Engineering of Multi-Agent Systems to Effectuate Distributed Data Mining Activities

Syed Zahid Hassan ZAIDI[a] , Syed Sibte Raza ABIDI[b] & Zafar Iqbal Hashmi[c]

[a,c]Health Informatics Research Group, School of Computer Sciences

Universiti Sains Malaysia, Penang, MALAYSIA

[b]Faculty of Computer Science, Dalhousie University, Halifax B3H 1W5, CANADA

Abstract

The proliferation of healthcare data has resulted in a large number of concerted efforts to inductively discover 'useful' knowledge from the collected data, and indeed interesting results have been reported by health informatics researchers. We argue that with the existence of multiple heterogeneous data repositories in a healthcare enterprise we need to establish a distributed data community, such that any DM effort draws upon the 'holistic' data available within the entire healthcare enterprise. When adopting this view, a set of data access and mining issues can be addressed using the well-known software agent technology. The aim of this paper is to propose the methodology of multi-agents system and engineering of individual agent to effectuate distributed DM activities applied to heterogeneous healthcare repositories.

Keywords:

Intelligent agents technology; Multi-agents organization; Agents' planning; and KQML.

1. Introduction

To date, many knowledge discovery systems have been developed prior to emergence of agent-based computing. The main function of these systems are to statically mine multiple level of knowledge from relational databases [1], but most of systems are not able to provide the services, pertinent towards the improvement/progress of any organization. The operational efficacy of an organization can be significantly increased by acquiring empirical knowledge from data repositories and by operationalzing procured empirical knowledge to derive the suite of decision support services that aim to impact strategic decision-making, planning and management of the organization [7]. The vantage point of these services are that they provide insights to assists healthcare analyst and policies makers to make strategic decisions or predict future consequences by taking into account the actual outcomes of current operative values. Typically, the services may include: *Analysis, planning, trending, examine, forecasting, predicting bench marking and best practices reporting, outcomes measurement, what–if scenario analysis, comparing organization practice with organization rules, market research , effectiveness on outcomes of treatment, data analysis for organization financing, health surveillance and resource allocations* [7].

To meet the above objectives, deployment of new analytic functionalities and data access methods have now become necessary for the systems to become the full fledged members of the new information era. Manually changing the systems is a nontrivial task. A preferable way of system working is to construct agent wrappers around KDD systems [3]. These agent wrappers interface to the information sources and information consumer, providing a uniform way to accessing data as well as offering additional functionalities, such as monitoring the changes and provide the services on demand. The critical question then is how to structure and organize these multiple agents to achieve user centric goal.

2. Multi-Agent System Methodology

There are number of methods have been proposed for the modelling of agents in a distributed heterogeneous environment [4] [6]. The best and widely used approach is to model agents based on BDI (believe, desires and intention) [4]. This approach looks the problem in two perspectives, the external and internal views. The external view break the problem into two main components: the agents themselves (agent model) and their collaboration/communication model (KQML). The internal view point use three models for each BDI agent class: an *agent model* for defining relationships between agents, *goal model* for describing goals, *planning and scheduling models* to achieve agent goal. In any distributed environment, the agents can be classified with particular roles according to their capability description [6]. Agents may have persistent roles— long term assignment as well as task specific role— short term assignments. In this point of view, we can comprise the multi-agents based organization into two main models: the *agent/role model* (agents' capability and behaviour) and the *agents/roles interaction model* (KQML). Moreover, the roles can be arranged in class hierarchy and the responsibilities are then assigned to each role along with services to meet those roles. Finally, agents' interaction can be defined down to the level of individual speech-acts and to associated data. To perform appropriate actions, a role can be defined with four general attributes: *responsibility, permissions, activities*, and *protocols* [6].

- *Responsibility;* agents/roles' functionality can be measured by its responsibility assigned to them which is divided into three categories: liveness property, safety and security properties. The liveness property ensures the "task will be done" by performing certain actions. For example, to illustrate it further, we discuss the monitoring responsibility of data collection agent/role. The common liveness property of that agent is to "inform the relevant agent in case of any updates in data resources". This property can be specified by using the liveness expression as,

  $$RoleName = Expression$$

  This liveness expression defines the 'life cycle' of the role and also constitutes either activities or protocol and both. In this context the data collection agent role might be

  $$DataMonitor = (Monitor.DataCollectionAgent, CheckStock. AwaitUpdate)$$

  This expression represents that DataMonitor consists of execution protocol Monitor, followed by the protocol DataCollectionAgent followed by the activity CheckStock and a protocol AwaitUpdate. In this case, the agent will definitely be required to ensure that the StockData is never empty, called its safety property.

  $$StockData > 0$$

- *Permissions* are the "rights" associated with the roles to realize their responsibility. Generally, those permissions are knowledge/information that the agents contain within its architecture. Return to DataMonitor role example, following is a simply illustration of the permission associated with the role DataMonitor:

  ```
  Read     supplied DataProvider   //data storage within the role (local info-base)
                    DataStatus     //updating
  Change            StockData      // level of data
  ```

This specification shows that the agent who carries out the data monitor role has permission to access, read and modify the data source.

- Roles' *activities* show it "private" action/computational functionality associated to them.

- Finally, *protocols* define the mechanism for the roles to interact or communicate each other.

As it depicted in figure 1, the DataMonitor role can be designed by writing role schema for each attribute associated with the role. Thus, by convention, role model can technically be defined; a role model is comprised of a set of role schemata, one for each role in the system.

Role	*DataMonitor*
Description: This role involves ensuring that database is kept on recorded and informs the agents in case of any updating.	
Protocols and Activities: *Monitor, DataCollectionAgent, CheckStock* and *AwaitUpdate*.	
Permissions:	
Reads	*supplied DataProvider* *DataStatus*
Changes	*StockData*
Responsibilities Liveness: *DataMonitor= (Monitor.DataCollectionAgent. CheckStock. AwaitUpdate)* Safety: *StockData > 0*	

Fig 1, shows schema for the role DataMonitor

3. Design of Agent Role model: Intelligent agent Framework

Researchers have presented numbers of diverse tasks deployment techniques to each agent in the MAS. In some cases, one agent can be model to perform only one particular task. This is a novice approach but increases system overhead as demands for more agents. The best approach is to aggregate all related-tasks and maps them with the particular agent type and the entire agent types could be controlled by single agent manager or supervisor agent [6]. This is actually a trade-off between the coherence of an agent types and the efficiency considerations that come into play when designing agent types.

Technically speaking, the classes of agents are quite similar to the objects in object oriented term. The main difference lies between them are; interfaces defined to the agents and the semantics of relationship between agents. The obvious agents' components include planner, scheduler, interface mechanisms, DM algorithms and rules to identify other agent type, discriminate 'agents' from traditional 'object' technology. The figure 2, demonstrates an 'abstract' high-level architecture which may vary somewhat with individual agents functionality within the system. The main building blocks of an agents' framework are the dispatcher, planner, scheduler, memory, conversations and execution monitor.

3.1 Conversation and Dispatcher Module

When an agent is started, the agent initialization module runs in its own thread. During initialization, it reads plan file— tasks and task reduction schemata, to process query/goal/objective from the user or any other agent. Once the agent has initialized, the control pass to the dispatcher module which waits for incoming KQML message— contains a KQML performatives and its associated information. Separate conversation module also

exists which maintain the state of agent's current conversation dialogue in accordance with a conversation policy defined.

3.2. Planner Module

It takes as input agent's current set of goals, the current set of task structures and a library task reduction schema. The agent planning process based on the hierarchal task network (HTN) planning formalism [2]. It consists of nodes that represent tasks and two types of arcs. The first type of arcs is Reduction links— that decompose the task into subtasks, and other represent Provision links—they are used for value propagation between task nodes. For example; the task t represents act of making the patient bill over his stay at hospital. The task t may decompose to find the cost t_1 of facilities provided to patient and t_2 maintaining the record.

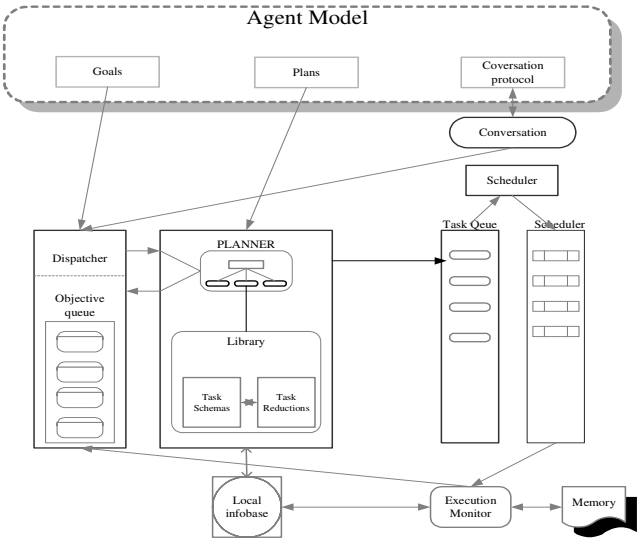

Figure 2, The Internal Architecutre of Intelligent Agent

3.2.1 *Task Representation*

A task is formally represented by a tuple <*N,Par,Dpar,Pro,Oout,C,E*>, where *N* is the name of the task. *Par* and *Dpar* are the static and dynamic parameters respectively that represent beliefs of the agent. These both parameters are stored in local InfoBase as a part of domain knowledge of the agent. The static parameter is known to the agent at a planning time and dynamic parameter is known only at an execution time of the plan. For example, as shown in fig 3, a plan to patient treatment in any particular branch of hospital, the origin and destination are static parameters while the patient disease (current status) and facilities available in the hospital are dynamic parameters since they are modified by the task in the plan. *Pro*, dynamic provision parameters, is a set of provisions are used to describe the set of flow of information in the plan. Provisions are used by the agent to run, to propagate the success or failure of a task in the plan. Provisions and dynamic parameters work combine together with outcomes, *Oout*— is a set of outcomes of the task. When an action is executed the resulting outcome values are transmitted through the provision links to the provisions or dynamic parameters of other task in the plan, establishing the execution conditions of later tasks. Finally *C* is a set of constraints which defines condition under

which task is to be executed and *E*, is a set of estimators used by the planner to predict the outcomes of the task.

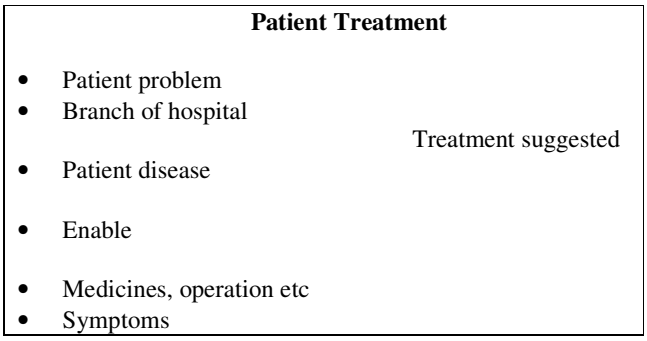

Figure 3, In this example, N = [patient treatment], Par = [Patient problem, branch of hospital], Dpar =[patient disease], Out = [treatment], C = [medicines, operation etc.], E = [symptoms]

3.3. Task Queue

The task queue is an interface between planner and scheduler used to store plans constructed by planner module. The task queue at any given time will contain the instantiated plans/ task structures (including all the actions and sub goals) that should be completed in response to any incoming request.

3.4. Scheduler Module

The agent scheduling process takes as input the agent's current set of task structures, particularly the set of all primitive actions, and decides which primitive actions and in what order need to be executed next.

3.5. Execution Monitoring and Memory Module

This process takes as input the agent's next intended action and prepares, monitors, and completes its execution by selecting plans from scheduling queue. Upon completion of any action, results are provided to the task network and the state of agent is stored in memory component in declarative form.

3.6 Local Infobase

The local infobase contains the information about the domain and changes its parameter w.r.t the environment changes. The description in local infobase also serves to generalize the services an agent offers to other agent.

4. Agents Communication Infrastructure

Multi-agents system contains number of heterogeneous intelligent agents to achieve user specific goal. The agents in distributed environment collaborate, corporate and interoperate among each other through one central middle agent.

In MAS, the agents can be categorized as services provider or services requester, based on its role description within an agent community. Service provider may contain the information and capability that services requester may desire. On the other hand, service

requester have a set of preferences for a particular types of parameters associated with desired service. Providers advertise their capabilities or services to middle agents and requesters submit requests to middle agent and select a provider according to their preferences.

Problem arise when a services requesters able to discover desired service provider through the middle agent but not able to communicate with it as being not having prior knowledge pertaining that agent capabilities and the format of messages. Agents communication language (ACL) such as KQML or FIPA [6], propose the adoption of common standard ACL and protocols to which all agent adhere. These languages specify the type of communicative action that the agents can perform as well as provide sender and other transport information regardless of providing specific contents of messages. Thus, Knowledge Query and Manipulation language (KQML) is a language that is designed to support interaction among high-level (with intentional description) software intelligent agents. It was developed by ARPA supported knowledge sharing effort (KSE), and successfully implemented and tested by several research groups and organizations. When using KQML, agents can exchange more complex objects in term of plans and goals or even share experiences. In addition, in KQML framework, Agents do not only engage in single-message passing but also they have conversations—shared sequence of messages that they follow. The message types of KQML are based on speech acts (KQML performatives), which in turn usually define beliefs, desires and intentional modalities (BDI theory).

Conclusion

Information has become the most valuable commodity in healthcare as the community is being overwhelmed with an influx of data that is stored in on-line distributed data repositories. Analyzing these data and extracting meaningful pattern demands the system having powerful analytical tool. The manual analysis of this data set is slow, expensive and highly subjective. In fact, such manual data analysis is becoming impractical as data volumes grow exponentially. In this paper, we have argued that agents-based data mining is abetting healthcare providers cut costs and improve care by automatically generating data mining healthcare reports that lead towards better plan from healthcare practitioners and policy makers regarding progress and improvement of healthcare organization. It does not only afford to provide productive medical interventions but also the outcomes data allow them to weed out the resources that aren't making conditions better.

References

[1] J. Han, Y. Fu, W. Wang, j. Chiang, W. Gong, K. Koperski, D. li, Y. lu, A. Rajan, N. stefanovic, B. Xia and O.R. Zaiane. DB Miner: A System For Mining Knowledge in Large Relational Batabases. In Int'l Conf. on Data Mining and Knowledge Discovery (KDD'1996), Portland, Oregon, August 1996, pages 250-255.

[2] M. Paolucci, O. Shehory, and K. Sycara, Interleaving Planning and Execution in a Multiagent Team Planning Environment, tech. report CMU-RI-TR-00-01, Robotics Institute, Carnegie Mellon University, January, 2000.

[3] Khaled A. Arisha, Fatma Özcan, Robert Ross, V.S. Subrahmanian, Thomas Eiter, Sarit Kraus, Impact: A Platform for Collaborating Agents (March/April 1999 (Vol. 14, No. 2), IEEE Intelligent Systems.

[4] Kinny, M. Georgeff, and A. Rao, A methodology and modelling technique for systems of BDI agents, Tech. Rep. 58, Australian Artificial Intelligence Institute, Melbourne, Australia, Jan 1996.

[5] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an Agent Communication Language," Software Agents, Menlo Park, Calif.: AAAI Press, 1997, pp. 291-316.

[6] K. Decker, K. Sycara, and M. Williamson. Modeling Information Agents: Advertisements,organizational Roles, and Dynamic Behaviour. In working notes of the AAAI-96 workshop on agent modeling, 1996.

[7] S.S.R. Abidi, Y-N. Cheah, A convergence of Knowledge management and data mining: towards 'knopwledge driven' strategic services, 3rd Int. Conferennce on the practical applications of knowledge management (PAKeM'2000), Manchester, 2000.