Interoperability and Semi-structured Data in an Open Web-based Agent Information System

Hongen Lu, Leon Sterling
Department of Computer Science and Software Engineering
University of Melbourne, Parkville, VIC 3052, AUSTRALIA
{helu,leon}@cs.mu.oz.au

Abstract

The World Wide Web is a vast resource of information and services that continues to grow rapidly. How to find information providers and how to integrate information agents in such an open environment are new challenges. In this paper we present an open multi-agent system, SportsAgents, for information gathering from the World Wide Web. In this system middleware agents are introduced to solve the interoperability problem among information agents. Novels in SportsAgents are an intelligent matchmaking algorithm for inferring the service relationships among information agents, and a Naive Bayesian model for information agents cooperation forming is given. Information agents use score patterns to extract semi-structured sports results. All these makes the system open and scalable to new applications and flexible to organise agents cooperation.

1 Introduction

Due to the explosive growth of the World Wide Web, finding specific information is becoming extremely difficult, sometimes even frustrating. To address this problem, several exciting new technologies have been developed. Search engines, such as Yahoo[1], AltaVista[2], and Lycos[3], are the most popular ones. They compile an index of the Web manually or automatically using "spiders". The index is used to answer users' queries based on keywords. The information providers based on search engines are built distributively over the Web using different languages and architectures. Most of them aim to provide users assistants for searching for interesting information.

However, the Web is a dynamic changing open environment with many new kinds of information providers becoming available daily. In such a dynamic environment information agents may appear and disappear at any time. Complete indexing of the Web is even more difficult, sometimes impossible [10]. So how to find the relevant information providers and how to integrate information agents are new challenges.

2 Middleware Agents

Intelligent information agents, such as Ahoy [4] [17], ShopBot [5] and SportsFinder [14], are programs that assist people to find specific information from the Web. We can view these agents as information service providers, which have the capabilities to find information for users, for example locating a person's homepage, finding the cheapest available prices for music CDs, or finding sports results of a team or a player.

However, for a novice user how to find these services, and for an information agent how to locate the service providers and communicate with them to solve its tasks cooperatively are challenges. One of the basic problems facing designers of open, multi-agent systems for the Internet is the connection problem — finding the other agents who might have the information or other capabilities that you need [3].

A possible solution is a software mediator. A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications [20]. In [2], a mediator is an information producing or serving entity in a large-scale internetworked environment.

3 SportsAgents System Architecture

SportsAgents is an open multi-agent system to answer a user's query about sports, for instance "which is the best

[1] http://www.yahoo.com
[2] http://www.altavista.com
[3] http://www.lycos.com
[4] http://ahoy.cs.washington.edu:6060

0-7695-0577-5/00 $10.00 © 2000 IEEE

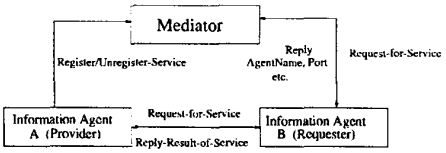

Figure 1. SportsAgents System Architecture

sports city in Australia?". In SportsAgents three types of agents, mediator, information agents and interface agents, are implemented. The domain of sports results is interesting as it highlights the contrast between the uniformity and diversity of information on the Web. For example, to find the best sports city may require the cooperation of different sports information agents.

In our architecture mediators are introduced to help information agents interact with each other. When an information agent starts, it can register its capability (what kind of service it can provide) with a mediator. When an information agent receives a query or a subtask within a query that can not be solved by itself, it can request the mediator to find out other agents that have the capability or a set of agents who can work cooperatively to provide that service. In Figure 1 the system architecture of SportsAgents is presented.

3.1 Interface Agents

An interface agent acts as an interface between the user and other information agents. It receives a user's query and makes a plan to answer this query, and then communicates with the mediator and other information agents to execute the plan. The result of the query is also presented to the user by the interface agent. In Figure 5, can start and quit an information agent via this GUI interface. The relevant debugging and receiving messages are also displayed to users on this interface.

3.2 Mediator

A mediator is a special kind of information agent acting as middleware to take as input, a request to find an agent that provides a service, and returns as output, a list of such agents and their cooperation relationships. A mediator also stores the services offered by different agents in the existing environment, and when a new agent is introduced into the environment it can register its capability to the mediator, if this agent wants its service to be used by others. Information agents also can unregister their services to the mediator when they want to quit the cooperation or exit. We present a mediator in SportsAgents in Figure 2.

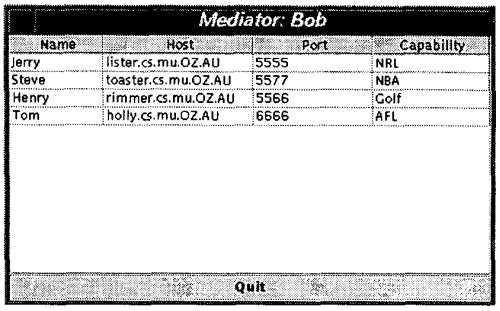

Figure 2. Mediator

3.3 Information Agents

Agent is one of the "hot" topics in the information age. The last ten years have seen a remarkable interest in agent oriented technology, spanning applications as diverse as information extraction, user interface and network management. Information agents are software programs that inhabit the world of computer events and networks to act on behalf of their human users to perform laborious information gathering tasks. Briefly an information agent is a program that has the ability to:

- express its capability to other agents, e.g. mediator.
- find certain information from the Web to answer user's query using its knowledge about a domain.
- communicate with other agents using an agent communication language, e.g. KQML.

The recent explosive growth of the World Wide Web provides a wealth of on-line information. This has made critical the need for some sort of intelligent assistant to a user who is searching for interesting information. Information agents are becoming popular due to this dramatic need.

In SportsAgents system, all the information agents are implemented using JACK. JACK is a Java-based multi-agent framework developed by Agent Oriented Software Pty. Ltd. It includes all components of the Java development environment as well as offering specific extensions to implement agent behaviour [11]. Following we give a segment code of an information agent:

```
agent InfoAgent extends Agent
{
  #handles event InfoEvent;
  #uses plan InfoPlan;
  #sends event RegiEvent rev;
  #sends event UnRegiEvent unrev;
  #sends event InfoEvent sev;
```

```
    #handles event UrlEvent;
    #uses plan SendPlan;
    #handles event RegiEvent;
    #uses plan ReactPlan;
    #posts event InfoEvent iev;
    #private database Url url("url.kb");

    public String Name, Host, Port,
                  Capability, Ontology;

    InfoAgent(String Name, String Host,
       String Port, String Capability,
       String Ontology)
      {.......}
    void sendto(String to, String cap)
      {.....}
    void register(String to, String
       name, String host, String port,
       String capability)
      {.....}
    void unregister(String to, String
       name, String host, String port,
       String capability)
      {.....}
    void find(String country)
      {.....}
}
```

The above blocks of codes give us a sense of an information agent's behaviours. When an information agent is instantiated, it will wait until it experiences an event that it must respond to. When such an event arises, it determines what course of action it will take. If the agent already believes that the event has been handled, then it does nothing. Otherwise, it looks through its plans to find those that are relevant to the request and then applicable to the situation. If it has any problems executing this plan, then it looks for others that might apply and keeps cycling through its alternatives until it succeeds or all alternatives are exhausted.

4 Semi-structured information extraction in SportsAgents

Sports results are usually published on the Web as semi-structured data. They highlight the contrast between the uniformity and diversity of information on the Web. Its great popularity and appeal mean that very few generalisations can be made about people who publish sports results on-line. Consequently, there is a wide variety of different formats, languages and convention used on sporting Web sites world wide. However, this range of formats is mitigated by the fact that the result of sporting contest is a very unambiguous piece of information, meaning that the agent can be presented with a very narrow request. Also, there are various cultural conventions which dictate how the scores should be displayed. Therefore, it can be seen that whilst the variety of formats in the sports score domain provide a great challenge, the uniformity provided by convention and the unambiguous nature of the results make the domain a plausible test bed for semi-structured information extraction.

Instead of a fully natural language understanding method, we use the score patterns to represent and extract the sports results. It just likes semi understanding of the text.

Definition 4.1 (Score Pattern) *A score pattern is an abstract expression of sports result using HTML tags and some special characters. It characterises the convention of that sport score.*

For example, the HTML source for a soccer result is:

```
<TR> <TD WIDTH=180> ARSENAL </TD>
     <TD WIDTH=50> 5 - 0 </TD>
     <TD WIDTH=180> BARNSLEY </TD>
</TR>
```

and we abstract its score pattern as:

```
Score Pattern::
  <TR > <TD * > TeamA </TD >
        <TD * > ScoreA - ScoreB </TD >
        <TD * > TeamB </TD >
  </TR >
```

Actually we use some special letters to represnet `TeamA`, `TeamB`, `ScoreA` and `ScoreB`. The whole alphabeta used for score pattern is the English alphabet plus some special signs, like $*$, -, and $<$.

So to extract a sport result, we first scan the HTML score segment into a pattern using the above alphabeta, let the pattern be $\mathcal{P} = p_1 p_2 \cdots p_n$; then we use the following function (1) to compare $\mathcal{P}$ with each score pattern $\mathcal{P}^k = p_1^k p_2^k \cdots p_m^k$ we have already known from previous sporting Web sites.

$$PosSim(\mathcal{P}, \mathcal{P}^k) = \frac{\max\limits_{0 \le i \le n, 0 \le j \le m} \{H_{ij}\}}{\sum\limits_{i=0}^{n} \sum\limits_{j=0}^{m} s(p_i, p_j^k) - W_{|m-n|}} \quad (1)$$

where $s(p_i, p_j^k)$ is the value matrix on score pattern alphabet and

$$H_{ij} = \begin{cases} H_{i-1,j-1} + s(p_i, p_j^k) \\ \quad \text{if } p_i \text{ and } p_j^k \text{ are associated} \\ H_{i-h,j} - W_h \\ \quad \text{if } p_i \text{ is at the end of a deletion of length } h \\ H_{i,j-l} - W_l \\ \quad \text{if } p_j^k \text{ is at the end of a deletion of length } l \end{cases}$$

We select the score pattern $\mathcal{P}^k$ with the maximum value of $PosSim(\mathcal{P}, \mathcal{P}^k)$ as a template to extract the sports score. For ladder sport results, like golf and cycling, we can compare the similarity between rows in a result table to find the standing or a team or player.

5 Service Matchmaking in SportsAgents

One function of a mediator is to provide the information, such as agent name, port, and capability to query agent, so that the query agent knows which agents to cooperate with to solve its problem. We call this process "*matchmaking*".

5.1 Service Hierarchy

Since each information agent is developed distributively, their capabilities are different from each other. SportsFinder [14] can find the sports results of golf, cycling, football and basketball etc. for users; while Ahoy [17] is good at locating people's homepages. However considering an application domain, such as sports, there exists a hierarchy relationship among these information agents. For example, information agent $\mathcal{A}$ can find all the results for Australian football teams, while agent $\mathcal{B}$ can only find the results of AFL (Australian Football League), in this case the service agent $\mathcal{B}$ can provide is a subset of agent $\mathcal{A}$, i.e. Service($\mathcal{B}$) $\subset$ Service($\mathcal{A}$).

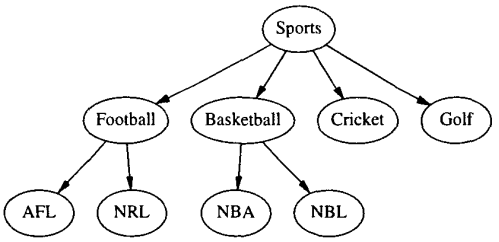

Figure 3. An Example of Sports Service Hierarchy

Definition 5.1 (Service) *A service* $S =< n, h, r, f >$ *is any computation provided by an information agent, where* n *is the name of this service;* h *is the host name of this information agent;* r *is the resource required* $< i, o >$ *which* i *is the input and* o *is output; and* f *is the functionality* S *can achieve, i.e.* $f : i \rightarrow o$.

In this section, we characterise agent service relations. We use S_i to denote the service of information agent $\mathcal{IA}_i$ and a set of sports names and/or competitions names to express what the service is about. For the above example, we have Service($\mathcal{B}$)={AFL}.

- **Identical Service:** $S_1 = S_2$.
- **Subservice:** $S_1 \subset S$.
- **Substitute Service:** a service S_1 can be substituted by service S_2, $S_1 \leftrightarrow S_2$.
- **Partial Substitute Service:** $S_1 \cap S_2 \neq \phi$.
- **Reciprocal Service:** $\exists S = S_1 \cup S_2 \wedge S_1 \cap S_2 = \phi$, then S_1 and S_2 are reciprocal with S.

To find the service relationship between two information agents, we need a data structure to describe the sports service hierarchy. We use a directed cyclic graph (DCG) to present the relations between agent services. The nodes in the graph present the services, and the edges are labelled with the service relationships. Following in Figure 3, we give a part of the sports service hierarchy.

5.2 Matchmaking Algorithm

Matchmaking is a process based on a cooperative partnership between information providers and consumers. In our SportsAgents system, a mediator is introduced to solve the connection problem, it finds the current available information agents who have the capability that the query agent (information consumer) is asking for. In case no available agent can fulfill the query service itself, the mediator will infer the available services to find a set of available information agents that can cooperate in some way to provide the query service.

We present the algorithm for service matchmaking in **Algorithm 5.1** .

6 Cooperation Forming in SportsAgents

Cooperation is often presented as one of the key concepts which differentiates multi-agent systems from other related disciplines such as distributed computing, object-oriented systems, and expert systems [6]. To form cooperation in an open multi-agent environment is extremely difficult. In SportsAgents we employ Naive Bayesian network to help information agents form cooperation with others.

A Bayesian network is a directed, acyclic graph that compactly represents a probability distribution. In such a graph, each random variable is denoted by a node. A directed edge between two nodes indicates a probabilistic influence from the variable denoted by the parent node to that of the child.

Definition 6.1 (Combined Service) *A combined service is a service that is achieved by a set of information agents working cooperatively. We use these information agents' service names and their cooperation relations to denote the combined service.*

Algorithm 5.1 matchmaking(S: Service, *head*: Hierarchy)

Given S, the service an information agent requests, agent table *AgentTable*, which contains the contact information of currently available information agents, and the service hierarchy *Head*. Each node in the service hierarchy has the following structure:

```
public class Service extends Vector
{
    String name;
    Service up;
    Service down;
    Service next;
        ⋮
}
```

this algorithm returns the agents contact information and their relationships.

find = false; head = Head;
Agent-found =null; relation = null;

if *AgentTable.index(S) = true* **then** {
  while *(AgentTable.query(S) != null)* {
    Agent-found = Agent-found
      ADD AgentTable.query(S);
  }
  relation = find-relation(S, head);
  find = true;
}
else {
  if *head.name* $\simeq$ *S.name* **then** {
    service = head.down;
    while *(service != null AND !find)* {
      matchmaking(S, service);
      service = service.next;
    }
  }
  else {
    matchmaking(S, head.down);
    matchmaking(S, head.next);
  }
}

return *Agent-found, relation.* □

Figure 4. Mathmaking Algorithm

For example, a combined service $CS = AND(S_1, OR(S_2, S_3), S_4)$ denotes the service that three information agents $\mathcal{IA}_1$, $\mathcal{IA}_4$, and one of $\mathcal{IA}_2$ and $\mathcal{IA}_3$ working together to provide. From the above definition we note that the result of our matchmaking algorithm is a set of combined services.

Following we give the decision making process when an information agent gets a set of combined services from the mediator, Naive Bayesian model is employed to choose a better combined service.

Suppose D is the decision to make, $D_k \in \{Accept, Reject\}$. Given a set of combined service $CS_1, \ldots, CS_n$, for each combined service CS_i we have p_i participating information agents $\mathcal{PIA}_i = \{\mathcal{IA}_{i1}, \ldots, \mathcal{IA}_{ip_i}\}$, and for each information agent $\mathcal{IA}_{ij}$ we select m features as $F(\mathcal{IA}_{ij}) = \{F_{ij}^1, \ldots, F_{ij}^m\}$.

From the Bayes theorem, the probability to make decision D_k under an instance cs of combined services is:

$$P(D = D_k | CS = cs) = \frac{P(CS = cs | D = D_k) P(D = D_k)}{P(CS = cs)} \quad (2)$$

We assume that each feature F_{ij}^h, $h = 1, \ldots, m$, is conditionally independent of every other features. This is the most restrictive assumptions embodied in the Naive Bayesian model. Formally this yields:

$$P(CS = cs | D = D_k) = \prod_{i=1}^{n} P(CS = cs_i | D = D_k) \quad (3)$$

For each $P(CS = cs_i | D = D_k)$, we have

$$P(CS = cs_i | D = D_k) = \frac{\sum_{j=1}^{|\mathcal{PIA}_i|} \lambda \sum_{h=1}^{m} P(F_{ij}^h | D = D_k)}{m \times |\mathcal{PIA}_i|} \quad (4)$$

where

$$\lambda = \begin{cases} 1 & \text{if } \mathcal{IA}_{ij} \text{ participates } AND \text{ in } CS_i \\ 1/q & \text{if } \mathcal{IA}_{ij} \text{ participates } OR \text{ in } CS_i \text{ with } q\ \mathcal{IA}s \end{cases}$$

Using the above model an information agent can evaluate the result it gets from the mediator and choose a set of promising service providers, whose probability of $P(Accept | CS = cs)$ is the greatest, to collaborate with. This process will improve the performance of single information agent.

7 Discussion

In [1], a platform called IMPACT to support multi-agent interactions is described. The platform provides a set of servers that facilitate agents interoperability in an application independent manner. In IMPACT, these servers match

the services within a given distance. The matchmaking algorithm does not consider the relationships among services.

The mediator in our architecture requires an arbitrary amount of deduction or knowledge to match any given service and request. Consider the example we give in Figure 2. Information agents Tom and Jerry can provide the service of extracting the results of AFL and NRL (National Rugby League) respectively. However to answer the user's query "which is the best sports city in Australia?", the agent must know the result of one of the most popular sports, Football, of Australia. Unfortunately, none of the current available information agents has that capability. Using our matchmaking algorithm, we can infer that Tom and Jerry can work collaboratively to get the information. So the mediator will reply the request information agent with Tom and Jerry's contact information as well as their cooperation relationship. This can not be achieved in IMPACT platform. We believe inference and reasoning abilities are necessary for mediators to provide intelligent matchmaking. We can build many domain specific mediators to help finding other information agents in the specific domain.

At this stage, we embed the agents contact details, such as agent names, port, and their cooperation relationships into agent communication language, which we use in SportsAgents is KQML. In the future, we will give a more complex and expressive agent capability description language for service advertising and requesting.

8 Conclusion

SportsAgents is an open mediator-based multi-agent system. In Figure 5, a result of SportsAgents is given. Mediator-based architecture is useful to build open Web-based applications, since in such a dynamic domain, applications are developed distributively and they become available or exist at any time. There is an obvious need for a standardised, meaningful communication among agents to enable them to perform collaborative task execution. Mediator-based architecture for Web applications is a step towards this goal. The programmer is responsible for defining the basic architecture of individual agents as well as the protocols governing their interactions. The functionalities of each agent may be simple. Complex behaviour of the whole multi-agent system emerges as dynamic interactions among its constituent agents. In such an architecture, new information agents can be easily incorporated into an existing multi-agent application. This makes the system open to new applications and flexible to organise agent cooperation.

The mediator in our architecture serves as middleware that not only solves the connection problem, but also infers the cooperation relationships among information agents, this will direct information agents to work in a cooperative way to answer user's query. Naive Bayesian model

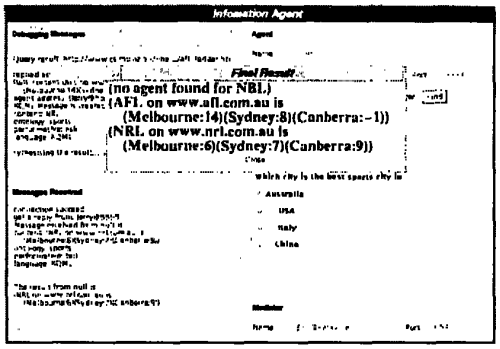

Figure 5. A Result of SportsAgents

is employed to help information agents to choose a better set of partners to cooperate with. In such a way, information agents can improve their capabilities, and information gathering from the Web becomes more scalable. Our future works are giving a more expressive agent capability description language, and improving the reasoning ability of mediator.

References

[1] K. Arisha, S. Kraus, V. S. Subrahmanian, and etal. IMPACT: Interactive maryland platform for agents collaborating together. *IEEE Intelligent Systems*, 14(2):64–72, March-April 1999.

[2] S. Dao and B. Perry. Information mediation in cyberspace: Scalable methods for declarative information networks. *Journal of Intelligent Information Systems*, 6(2/3):131–150, May 1996.

[3] K. Decker, K. Sycara, and M. Williamson. Matchmaking and brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, December 1996.

[4] K. Decker, K. Sycara, and M. Williamson. Middle-Agents for the Internet. In *Proceedings of 15th IJCAI*, pages 578–583, Nagoya, Japan, August 1997.

[5] R. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the World Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, 1997.

[6] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3):309–314, 1997.

[7] E. C. Freuder and R. J. Wallace. Suggestion strategies for constraint-based matchmaker agents. In *The AAAI-97 Workshop on Constraints and Agents*, 1997.

[8] D. Kuokka and L. Harada. Supporting information retrieval via matchmaking, 1995.

[9] D. Kuokka and L. Harada. Integrating information via matchmaking. *Journal of Intelligent Information Systems*, 6(2/3):261–279, May 1996.

[10] S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.

[11] A. O. S. P. Ltd. *JACK Intelligent Agents User Guide*, 1999.

[12] H. Lu and L. Sterling. A mediator-based multi-agent architecture for information gathering from the Web. In J. Grundy, editor, *Proceedings of the 2nd Australian Workshop on Software Architectures*, pages 109–117, Melbourne, Australia, November 1999. in conjunction with TOOLS Pacific'99.

[13] H. Lu, L. Sterling, and A. Wyatt. Knowledge discovery in SportsFinder: An agent to extract sports results from the Web. In N. Zhong and L. Zhou, editors, *Methodologies for Knowledge Discovery and Data Mining, Third Pacific-Asia Conference (PAKDD-99) Proceedings*, volume 1574 of *Lecture Notes in Artificial Intelligence*, pages 469–473, Beijing, China, April 1999. Springer.

[14] H. Lu, L. Sterling, and A. Wyatt. SportsFinder: An information agent to extract sports results from the World Wide Web. In *Proceedings of the Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM99)*, pages 255–265, London, UK, April 1999. The Practical Application Company Ltd.

[15] S. Saeyor and M. Ishizuka. Cooperative matchmaking of requests among distributed change monitoring service agent. In *1999 IEEE Pacific Rim Conference on Communications, Computer and Signal Processing (PACRIM'99)*. University of Victoria, Victoria, B.C., Canada, August 1999.

[16] T. Sandholm. eMediator: a next generation electronic commerce server. Technical Report WUCS-99-02, Department of Computer Science, Washington University, St. Louis, MO 63130, January 1999.

[17] J. Shakes, M. Langheinrich, and O. Etzioni. Dynamic reference sifting: A case study in the homepage domain. In *Proceedings of the Sixth International World Wide Web Conference*, pages 189–200, 1997.

[18] K. Sycara, J. Lu, and M. Klusch. Interoperability among heterogeneous software agents on the Internet. Technical Report CMU-RI-TR-98-22, the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, October 1998.

[19] K. Sycara, A. Pannu, M. Williamson, D. Zeng, and K. Decker. Distributed intelligent agents. *IEEE Expert, Intelligent Systems and their Applications*, 1996.

[20] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), March 1992.

[21] M. Wooldridge and N. R. Jennings. Cooperative problem-solving. *Journal of Logic and Computation*, 9(4):563–592, 1999.