Design and Implementation of the Agent-based EVMs System

By

Hui-Min Chen
June 30, 2000

This report describes the design and implementation of the agent-based EVMs system. A prototype is available at *www.sims.berkeley.edu/research/metadata/demo.html.*

1.0 Intelligent Agents

Since 1990, intelligent agents have been broadly used in complex, dynamic, and open applications such as production planning, robotics, and in searching the Internet. Though theoretical research as well as real implementation of agent technology are common, there is no commonly agreed-upon definition of intelligent agents. From an Artificial Intelligence point of view, an intelligent agent is a hardware or (more usually) software-based computer system that has the following properties: (Wooldridge & Jennings 1995)

- Autonomy: agents work on their own without direct interventions of humans or others.
- Social ability: agents interact with each other using an agent communication language (ACL, e.g., Telescript, Safe-Tcl, KQML, etc.).
- Reactivity: agents perceive their environment and respond in a timely manner.
- Pro-activeness: agents are goal-oriented.

Besides, Franklin & Grasser (1996) suggested that an intelligent agent should be temporally continuous, i.e., a continuously running process, and may have a learning ability and mobility, i.e., transporting itself from one machine to another. Haverkamp & Gauch (1998) suggested that an intelligent agent is benevolent (not a Trojan horse) and behaves in a rational manner. For other definitions of intelligent agents, see Franklin & Grasser (1996) for detail.

In an agent-based system, there is a single agent such as an interface agent or multiple agents. Generally speaking, a multi-agent-based system has one of the following three types of architecture (Genesereth & Ketchpel, 1994): a contract-net system, a specification-sharing system, or a federated system. In a contract-net system, agents in need of service broadcast requests for proposals to other agents. Upon receiving requests, agents offer bids to the originating agents based on their capabilities. The originating agents then decide which agents to deal with for contracting. The communication cost of a contract-net system is $O(N^2)$, where N is the number of agents. In a specification-sharing system, agents supply other agents with information about their capabilities and needs, and, therefore, agents can coordinate their activities based on a mutual understanding. The communication cost of a specification-sharing system is

between $O(N)$ and $O(N^2)$. Both a contract-net system and a specification-sharing system utilize a direct communication approach.

In contrast, a federated system has a hierarchical structure in which a coordinator or a facilitator supervises local agents in a federation (Geneserth & Fikes, 1992). The federated agents do not communicate with each other directly. Instead, they communicate only with the coordinator to request a service from agents in other federations. The coordinators know each local agent very well; therefore, they broadcast the needs of their local agents and pass only relevant service requests from other coordinators to their local agents. One benefit of such an indirect communication approach is that communication cost can be reduced to a minimum. That is, $O(M^2)$, $M << N$, where M is the number of federations.

An agent-based system could be very complex and thus expensive. What kind of problem justifies the use of agent technologies rather than other software techniques such as object-oriented programming or pure artificial intelligence application? Muller (1998) provided guidelines in answering this question. He suggested that agent technologies are appropriate for application problems having the following properties:

1. Highly dynamic, necessary to be responsive and adaptable to a changing environment.
2. Need to deal with failure, e.g., rescheduling, re-planning, and reallocating of resources.
3. Need to balance long-term, goal-directed, and short-term, reactive behavior.
4. Complex and/or safety-critical guaranteed reactions and response time.
5. Geographically or logically distributed resources.
6. Need for reliability, robustness, and maintainability.
7. Complex or decentralized resource allocation problems with incomplete information.
8. Flexible interaction with human users.

According to Muller's guidelines, it is quite evident that agent technologies are suitable for the EVMs system for the following reasons:

1. The EVMs system intends to provide a search aid for users to deal with unfamiliar databases on the Internet. The Internet is geographically distributed and is known for its highly dynamic nature.
2. The metadata vocabulary of an unfamiliar database may be very complex and change over time. Therefore, a system that can respond to and adapt to a changing environment is desirable.
3. Remote, unfamiliar databases may be unavailable temporarily due to remote server down or network problems and so the EVMs system should have an ability of rescheduling or re-planning its tasks.
4. The entry vocabulary module building process should be reliable and robust. Also a decomposition of such a complex task is preferable to facilitate the whole process.

5. Most importantly, the entry vocabulary module building process should be transparent to users. An advanced EVMs system will interact flexibly with users in the front-end, collect their queries, detect their preferences or domains of interest, and create entry vocabulary modules behind the scene.

2.0 Related Work

The application of intelligent agents to the design of information retrieval systems has drawn some attention in recent years and their benefits have been demonstrated in several publications. For example, SAIRE (A Scalable Agent-Based Information Retrieval Engine) developed by Odubiyi et al. (1997) is a multi-agent search engine that uses agent technology, natural language understanding, and conceptual search techniques to support public access to Earth and Space Science Data over the Internet. SAIRE provides a Web-based, integrated user interface to distributed data sources maintained by NASA and NOAA. Agents in SAIRE form seven federations, in which an agent manager controls several local agents. However, agent managers do not intercommunicate with each other directly but rather via a central coordinator agent (master agent). This hierarchical design is risky because the whole system is down if the central coordinator fails. Moreover, a natural language parser extracts a frame containing actions and concepts embedded in the user's natural language input. Two dictionaries, a main dictionary and a personal dictionary, are used to specify a user's domain of interest. The main dictionary contains words with semantic meaning related to SAIRE's specific domains, while the personal dictionary contains words that might have multiple meanings in the domain, as well as new words defined by users. Thus, SAIRE can learn new words through interacting with users to define unknown words as well as to clarify words with multiple meanings.

Amalthaea (Moukas & Zacharia 1997) is a multi-agent ecosystem for personalized filtering, discovery and monitoring of information sites. Its main goal is to assist the users in finding interesting information on the Web. There are two kinds of agents in Amalthaea: filtering agents that model and monitor the interests of the user and discovery agents that model the information sources. Both the user's interest and retrieved documents from Web sites are represented by weighted keyword vectors. The information agents pick one document from the downloaded set passed by the discovery agents and calculate how a confidence level is that specific document will satisfy the user's needs. The confidence measure is not different from the typical normalized similarity measure (cosine) used in the vector space model in information retrieval. A particular feature of Amalthaea is that it provides a market-like ecosystem in which agents evolve, collaborate and compete to survive. Agents that are valuable (useful) to the user and to other agents are allowed to reproduce while low-performing agents are destroyed to save system resources. Though its multi-agent architecture was not explicitly specified, Amalthaea appeared to be a kind of specification sharing systems.

3.0 Architecture of the EVMs System

Running on a UNIX platform, the EVMs system is implemented in the conventional TCP/IP client-server environment. When a user makes a connection to the EVMs Web site at *www.sims.berkeley.edu/research/metadata/,* the Web server initiates a session (channel) between the user and the EVMs system via a Common Gateway Interface (CGI). The external architecture of the EVMs system is shown in Figure 1.

To reduce communication cost as well as system workload, the EVMs system utilizes a federated system (see Figure 2) consisting of multiple interacting agents of eight types which "divide and conquer" the entry vocabulary module building tasks. They range from data retrieval agents (type 4 in Figure 2) that download records from remote databases through cleaner agents (type 5 in Figure 2), extractor agents (type 6 in Figure 2), and builder agents (type 7 in Figure 2) that create a dictionary of associations using downloaded records. Desktop agents (type 1 in Figure 2) and domain agents (type 2 in Figure 2) help the user define the domain of interest and deploy the created association dictionaries. Planner/scheduler agents (type 3 in Figure 2) schedule and reschedule record downloading tasks in a near-optimal way. The agents (facilitators) communicate with each other using the KQML (Knowledge Query and Manipulation Language, see *www.cs.umbc.edu/kqml* for details)-like agent communication language (ACL), which consists of tagged messages. For example, an ACL message sent from the desktop agent to the planner/scheduler agent to request an estimation of the record downloading time is as follows:

```
<message>
    <msg id>service_type_3</msg id>
    <usr id>20000620_2_11:32:56</usr id>
    <sender>desktop</sender>
    <receiver>planner</receiver>
    <database>inspec</database>
    <domain>information_systems</domain>
    <service requested>estimate</service requested>
    <reply constraint>urgent</reply constraint>
</message>
```

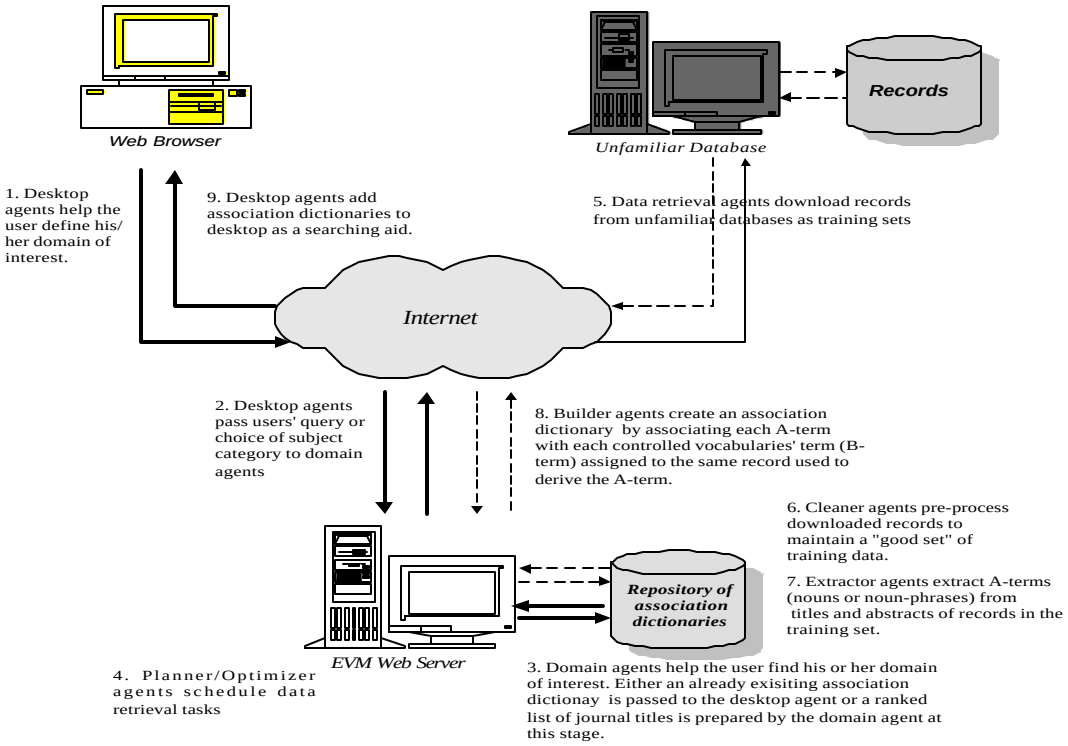

Figure 1. Architecture of a Multi-Agent-Based Search Support for Unfamiliar Databases (EVMs)

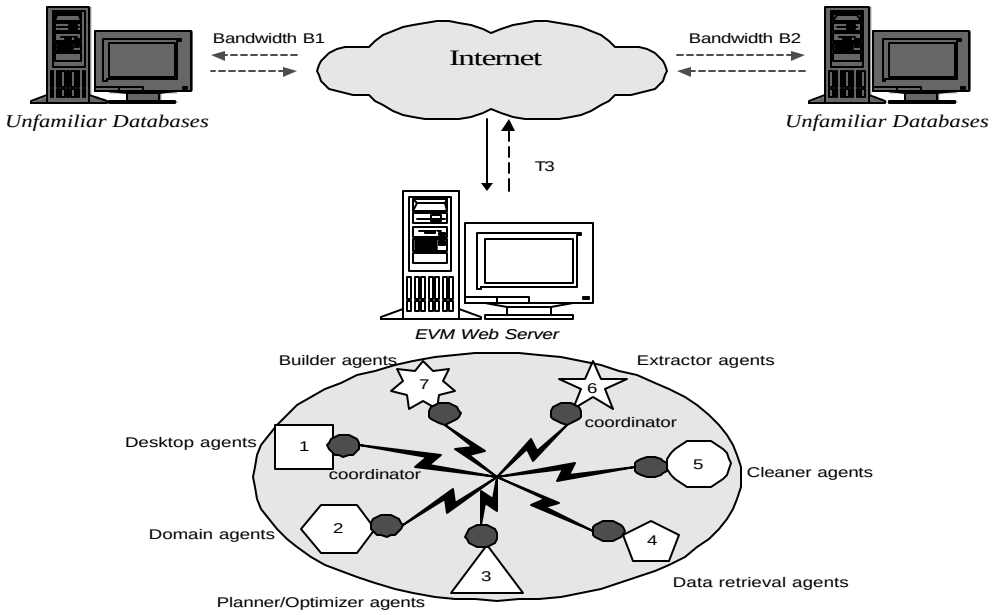

Figure 2. Internal Archiecture of the EVMs system

Whenever a session is initiated, a federation is formed and identified by a unique ID composed of the session initiation time and the user's personal identification number (will be introduced in the following section). Similarly, a federation is collapsed when the corresponding session ends.

The EVMs agents are introduced below in the order in which they come into play during a session.

3.1 Desktop Agents

The desktop agent provides a graphical interface that links the user with other agents. It is the only type of agent that resides on the client computer to communicate with the user directly. The major tasks of the desktop agent are to:

1. Authenticate the user's identity. Each new user is invited to create a personal profile consisting of the user's email address as well as a unique alphanumeric personal identification number (PIN) up to six characters. For every returned user, a valid PIN is required to retrieve the archive of existing entry vocabulary modules or create a new entry vocabulary module. The PIN helps to recover the user's activities from transaction logs for deriving user preference.
2. Examine the validity of the user's input.
3. Provide online help to the user.
4. Work with the domain agent to help the user define a domain (subject) of interest.
5. Acquire the status of other agents and display this information on the desktop in a real-time fashion (see Figure 3).
6. Add the resulting entry vocabulary modules onto the desktop (see Figure 4) to assist the user searching the database (see Figure 5). Some of these databases are restricted by licensing conditions and accessible only to users with a Berkeley IP address.
7. Detect the termination of a session due to user inactivity.

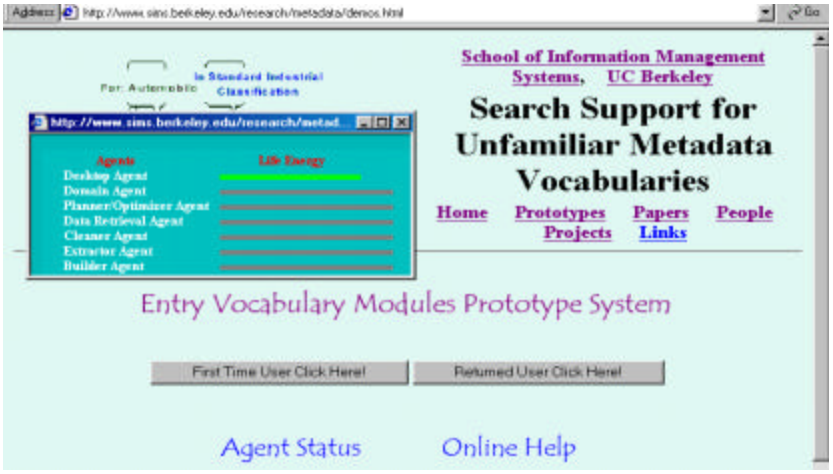

Figure 3. Check agent status "on-the-fly".

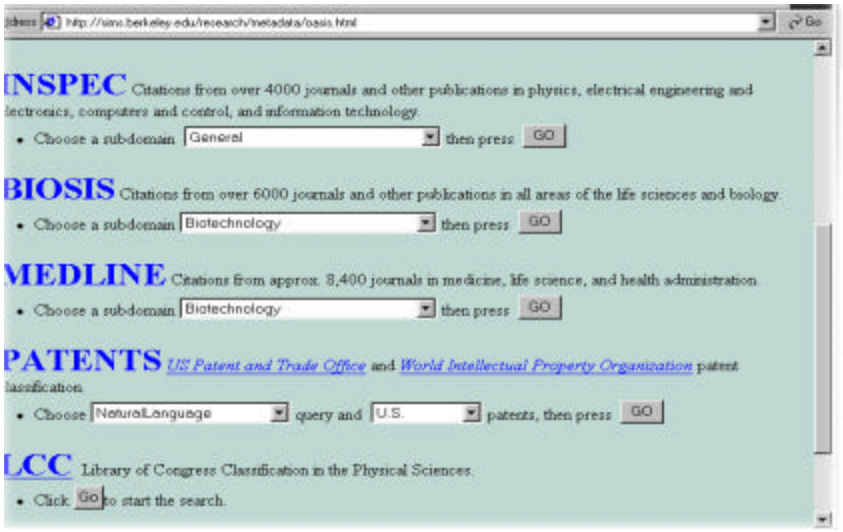

Figure 4. Add dictionaries to the desktop.

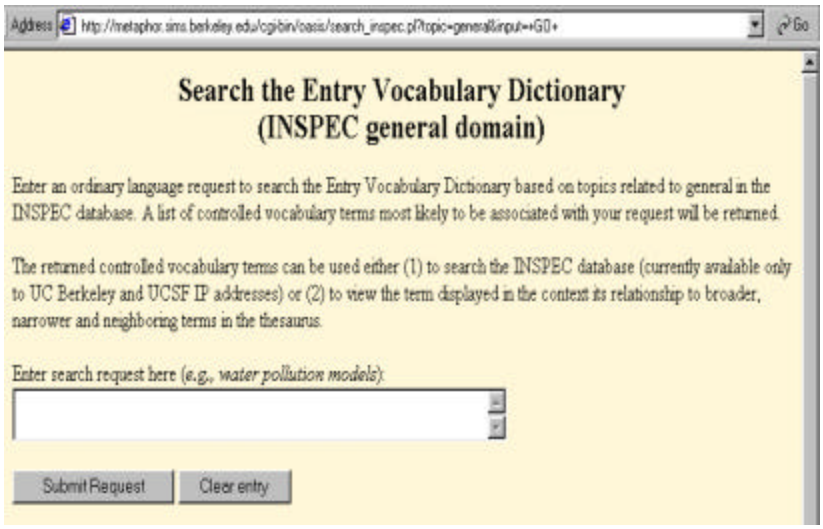

Figure 5. Search the dictionary for INSPEC Thesaurus terms.

3.2 Domain Agents

The task of the domain agent is to define the user's domain of interest. This is implemented by the preparation of a ranked list of journal titles from the Sciences Citation Index (SCI) *Journal Citation Report* that reflect the user's domain of interest. Ordinarily, the metadata vocabulary should be studied as a whole. However, in practice, users are rarely equally interested in all of the contents of the whole database at any one time. Instead, they are usually interested in some specific domain reflecting their

particular interest in a search. Thus, it will be more efficient and cost-effective to concentrate on entry vocabulary modules of topical domains within the database.

The domain agent will first ask the user to choose a database to search. Then, the user has to select one of the subject categories for the chosen database (see Figure 6). The domain agent will search the archive of entry vocabulary modules to see if one (of the same subject category for the same database) already exists. If yes, the domain agent just passes it to the desktop agent directly. Otherwise, a ranked list of journal titles is prepared as queries by the domain agent and passed to the data retrieval agent to retrieve records from the chosen database. Meanwhile, the desktop agent passes a request for creating an entry vocabulary module to the planner/scheduler agent as well.

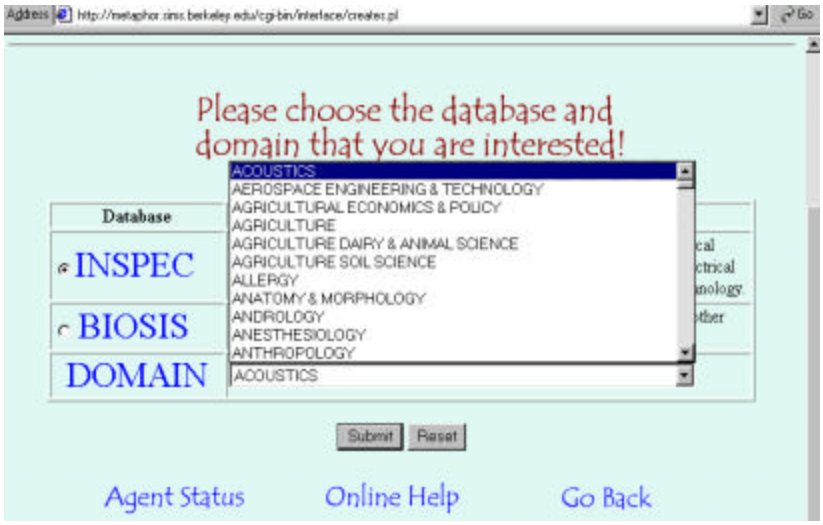

Figure 6. Choose a domain of interest.

3.3 Planner/Scheduler Agents

In general, data retrieval (downloading) might be extremely time-consuming and thus, becomes a bottleneck. Therefore, the primary purpose of the planner/scheduler agent is to schedule the data retrieval tasks so that the EVMs system is guaranteed to work smoothly and efficiently. The planner/scheduler agent adopts a heuristic rule (called LCT or least completion time) to generate a near-optimal schedule for the tasks. The principle of the LCT rule is that the task having the shortest expected completion time is to be processed first during peak hours. A peak hour is defined as the time at which the number of tasks is more than a threshold value Z, which is defined as the largest integer less than C/2B, where C is the number of communication channels provided by the cable line connecting the EVMs system to the Internet and B is the average number of communication channels consumed by a data retrieval task. For a T3 line (currently used by the University of California, Berkeley), there are 672 separate communication channels, each channel with a data transmission rate of 64,000 bits per second (Thomas, 1996). If the average data

transmission rate is 300,000 bits per second, then B = 5 (300,000/64,000) and therefore, Z = 67 (672/10). During regular hours, the tasks are served on a first-come, first-served basis.

To estimate the completion time (including downloading time and processing time) of a task, the planner/scheduler agent employs a multiple regression model as follow:

$$T = \beta_0 + \beta_1 R + \beta_2 N \qquad (1)$$

where T is the expected completion time of a task, R is the number of records (articles) to be downloaded, and N is the number of tasks being processed. This is because records are downloaded and processed in sequence; therefore, the completion time is positively associated with the number of records. To reflect changes in the size (bytes) of records and data transmission rates associated with remote databases, the regression parameters are updated every week using the most recent two weeks' historical data. As of the time when this paper was written, β_0 = -600.45, β_1 = 0.16, β_2 = 274.53, R-square = 0.89 (proportion of variances in T that are accounted for by R and N), and the average completion time of a task is 25 to 59 minutes, depending on the size of records.

Once the schedule for a task is determined, the planner/scheduler agent will notify the user by email about the expected completion time of his or her request. If a downloading or processing task is interrupted due to network down or system malfunction, the planner/scheduler agent will reschedule the task, update the expected completion time by rerunning (1), and notify the user about this change.

3.4 Data Retrieval Agents

Data retrieval agents are designed to download records from the unfamiliar database. Those records serve as the training set to create an entry vocabulary module. Records used to build entry vocabulary modules are typically acquired by downloading sets of MELVYL ® records retrieved by a query consisting of the ranked list of journal titles prepared by domain agents (An exception to this approach is the U.S. patents records that are retrieved from the U.S. patent database online.). MELVYL ® is an online catalog of the University of California and provides a uniform interface to a variety of databases such as INSPEC, BIOSIS, SIC, and MEDLINE (*www.melvyl.ucop.edu*).

3.5 Cleaner Agents

The multiplicity of formatting, even with SGML formatting of text, implies a difficult and demanding job of standardizing formats to be understood and prepared for processing by other agents. The records in the downloaded set must be transformed or filtered by the cleaner agent before the entry vocabulary module can be constructed.

3.6 Extractor Agents

For each record in the training set, the extractor agent extracts terms (nouns or noun-phrases) in the title and abstract using natural language processing modules. These natural language terms are called A-terms. Then each A-term is paired with the metadata terms, called B-terms, assigned to that record. The extractor agent ends with a list of A-term and B-term pairs for each record in the training set.

The natural language processing modules consist of two publicly available applications: the Apple Pie Parser (Version 5.9) and the Brill Tagger. They are mainly employed to identify short noun phrases in a sentence. This is because noun phrases are more likely than single words to be used to represent complex concepts. The Apple Pie Parser is a bottom-up probabilistic chart parser that looks for the parser tree with the best score using a best-first algorithm. The Brill Tagger is a transformation-based part of speech tagger and has been applied to experiments investigating how WordNet can be used with contextual clues to disambiguate terms (Leacock & Chodorow, 1998). For a detailed description of the application and evaluation of the Apple Pie Parser and the Brill Tagger, see Kim & Norgard (1998).

3.7 Builder Agents

The major task of the builder agent is to compute the association between the A-term (words or noun phrases) and the B-term (metadata terms) in a pair following (2)-(5) to complete an entry vocabulary module. Once an entry vocabulary module is complete, the builder agent either forwards it to the desktop agent if the user is still online, or adds it to an archive and notifies the user by email.

$$f_0 = C_{|A\cap B|}^{|B|} p^{|A\cap B|}(1-p)^{|B|-|A\cap B|} * C_{|A\cap \sim B|}^{|B|} p^{|A\cap \sim B|}(1-p)^{|\sim B|-|A\cap \sim B} \quad (2)$$

$$f_1 = C_{|A\cap B|}^{|B|} p_1^{|A\cap B|}(1-p_1)^{|B|-|A\cap B|} * C_{|A\cap \sim B|}^{|B|} p_2^{|A\cap \sim B|}(1-p_2)^{|\sim B|-|A\cap \sim B|} \quad (3)$$

$$\lambda = f_0 / f_1 \quad (4)$$

where $p = (|A\cap B| + |A\cap \sim B|)/(|B| + |\sim B|)$, i.e., P(A|B) = P(A|~B) = P(A) under the null hypothesis that term A (a A-term) and term B (a B-term) are independent $p_1 = |A\cap B|/|B|$ and $p_2 = |A\cap \sim B|/|\sim B|$ are the maximum likelihood estimators (MLEs) of P(A|B) and P(A|~B) respectively. Moreover, (4) can be transformed into the following logarithmic format:

$$\begin{aligned} -2\ln\lambda = 2[&|A\cap B|\ln p_1 + (|B| - |A\cap B|)\ln(1-p_1) + \\ &|A\cap \sim B|\ln p_2 + (|\sim B| - |A\cap \sim B|)\ln(1-p_2) - \\ &|A\cap B|\ln p + (|B| - |A\cap B|)\ln(1-p) - \end{aligned}$$

$$| A \cap \sim B | \ln p + (| \sim B | - | A \cap \sim B |) \ln(1 - p)] \quad (5)$$

Now $-2 \ln l$ can be interpreted as the "weight" or "strength" of associations between term A and term B. For example, the contingency table for term A "copyright" and the INSPEC Thesaurus term (term B) "legislation" is shown as Table 1. Plugging these numbers into (5) will obtain the "weight" of associations between "copyright" and "legislation", which is 285.25.

Table 1. The contingency table for the A-term "copyright" and the B-term "legislation" in INSPEC.

	B-term	**Non-B-term**	**Subtotal**
A-term	45	268	313
Non-A-term	2524	1043845	1046369
Subtotal	2569	1044113	1046682

4.0 Reference

Franklin, S. & Grasser, A. (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *In Proceedings of the Third International Workshop on Agent Theories*, Archietctures, and Languages, Springer-Verlag.

Genesereth, M. R. & Fikes, R. E. (1992). An agent-based approach to software interoperability. *In Proceedings of the DARPA Software Technology Conference*, 359-366, Meridian Corporation, Arlington, VA.

Genesereth, M. R. & Ketchpel, S. P. (1994). Software agents. *Communication of the ACM*, 37(7), 48-53.

Haverkamp, D. S. & Gauch, S. (1998). Intelligent Information Agents: Review and Challenges for Distributed Information Sources. *Journal of the American Society for Information Science*, 49(4), 304-311.

Kim, Y. & Norgard, B. (1998). Adding natural language processing techniques to the Entry Vocabulary Module building process. *Technical report*.
Available :*www.sims.berkeley.edu/research/metadata/nlptech.html.*

Leacock, C. & Chodorow, M. (1998). Combining local context and Wordnet similarity for word sense identification. In C. Fellbaum, editor, WordNet: *an electronic lexical database*. MIT Press, Cambridge, MA.

Moukas, A. & Zacharia, G. (1997). Evolving A Multi-Agent Information Filtering Solution in Amalthaea. *In Proceedings of the First International Conference on Autonomous Agents*, February 5-8, 394-403.

Muller, J. P. (1998). Architectures and applications of intelligent agents: a survey. *The Knowledge Engineering Review*, 13(4), 353-380.

Odubiyi, J.B., Kocur, D.J., Weinstein, S.M., Wakim, N, Srivastava, S., Gokey, C., & Graham, J. (1997). SAIRE- A Scalable Agent-Based Information Retrieval Engine. *In Proceedings of the First International Conference on Autonomous Agents*, February 5-8, 292-299.

Thomas, R. M. (1996). An introduction to local area networks. Alameda, CA: SYBEX.

Wooldridge, M & Jennings, N. R. (1995) Agent Theories, Architectures, and Languages: a Survey. In Wooldridge and Jennings Eds., Intelligent Agents, Berlin:Springer-Verlag.