

Collaborative Kodama Agents with Automated Learning and Adapting for Personalized Web Searching

Tarek Helmy Satoshi Amamiya Makoto Amamiya

Department of Intelligent Systems

Graduate School of Information Science and Electrical Engineering

Kyushu University

6-1 Kasugakoen, Kasuga-shi

Fukuoka 816-8580, Japan

E-mail: [helmy,roger,amamiya]@al.is.kyushu-u.ac.jp & Fax: 81-92-583-1338

Abstract

The primary application domain of Kodama¹ is the World Wide Web and its purpose in this application is to assist users to find desired information. Three different categories of Kodama's agents are introduced here, Web Page Agents (WPA), Server Agents (SA), and User Interface Agents (UIA). Kodama agents learn and adapt to the User's Preferences (UP), which may change over time. At the same time, they explore these preferences to get any relevancy with the future queries. The main trust of Kodama research project is an investigation into novel ways of agentifying the Web based on the pre-existing hyper-link structure. These communities of Kodama agents automatically achieve and update their Interpretation Policies (IP) & UP and cooperate with other agents to retrieve distributed relevant information on the Web. We focus in this paper on the implementation and the evaluation on the adaptability of Kodama agents with the UP. This paper proposes a new method for learning the UP directly from user's interaction with the system and adapting the preferences with user's responses over the time. The user's feedback is used by the Kodama to support a credit adaptation mechanism to the IP of the WPA that is responsible for this URL and to adapt the weight and the query fields in user's query history and bookmark files. In terms of adaptation speed, the proposed methods make Kodama system acts as a PinPoint information retrieval system, converges to the user's interests and adapts to the sudden change of user's interests over time.

Introduction

The number of information sources available to the Internet user has become extremely large. This information is loosely held together by annotated connections, called hyperlinks [Kleinberg, 1999], [Chakrabarti *et al.*, 1998]. This information abundance makes increasing the complexity of locating relevant information. The model behind Traditional Search Engines (TSE) analyzes collected documents once to produce indices, and then amortizes the cost of such processing over a large number of queries, which access the same index. This model

assumes that the environment is static. The Web is however highly dynamic, with new documents being added, deleted, changed, and moved all the time [Menczer and Monge, 1999]. At any given time an index of the TSE will be somewhat inaccurate (e.g., containing stale information about recently deleted or moved documents) and somewhat incomplete (e.g., missing information about recently added or changed documents). Also users normally face with very large hit lists with low precision. Moreover, the information gathering and retrieving processes in TSE are independent of user's preference, and therefore feedback from the later process is hardly adaptive to improve the quality of the former process. These factors make it necessary to investigate new techniques to address such problems. Intelligent agents may be the way to improve search and retrieval process as active personal assistants. The combination of the search engine, the agent, the UP algorithm, and the Information Retrieval (IR) algorithm addresses the trust and competence issues of agents.

Researchers in Artificial Intelligence (AI) and IR fields have already succeeded in developing agent-based techniques to automate tedious tasks and to facilitate the management of information flooding [Pann and Sycara, 1996], [Chen and Sycara, 1998], [Edmund *et al.*, 2000]. A way to partially address the scalability problems posted by the size and dynamic nature of the Web is to decentralize the index-building process. Dividing the task into localized SAs that agentify specific domains by a set of WPAs developed in this project. The success of this project has been achieved by the cooperation among the WPAs [Helmy *et al.*, 2000a], [Helmy *et al.*, 1999b], [Helmy *et al.*, 1999c]. The approach is based on a distributed, adaptive and on-line agent population negotiating and making local decisions for retrieving the most relevant information to the user's query.

In this paper we will start by describing the architecture and the communication of Kodama agents. Next, we describe the mechanism of agentifying a Web site. We then discuss our new methodologies of calculating the relevancy of retrieved Web page contents to the UP, which is used in UIA and WPA. We focus on the

¹ Kyushu university Open Distributed Autonomous Multi-Agent

* Copyright © 2001, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

implementation and evaluation on the adaptability of Kodama agents with the UP. We also describe the learning techniques used in WPA and UIA. Finally we present the experimental results and future work of Kodama system.

Kodama Architecture and Communication

Figure 1 illustrates the architecture of Kodama agent. Each agent draws upon a sophisticated reasoning architecture that consists of different reusable modules. We divided an agent into a kernel unit and an application oriented unit. The former unit contains standard modules, data structures, and methods for communications, message interpretations, coordination and learning provided by Kodama, and the latter is defined by a designer and contains inter-agent specific communications, interpretations and processes. Kodama framework is being used to develop distributed collections of intelligent agents that cooperate asynchronously to perform goal-directed information retrieval and integration. Kodama system allows agents to find each other by providing a mechanism for registering each agent's capabilities. Each agent must be able to interpret the input message sent to it from other agents if the performative field of the message requests the agent to do it. If an agent is unable to interpret some input, it consults its down-chain agents. The software designer has been responsible for providing each agent with its IP so far [Hodjat and Amamiya, 2000]. In Kodama this is done automatically by the WPA itself. Kodama agent actuates other modules based on the messages received from other agents. Each message is comprised of a message content, and a performative that specifies what should be done with that content.

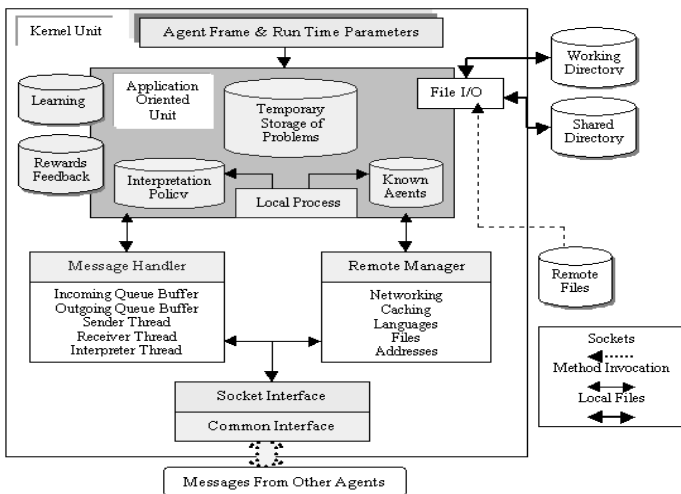


Figure 1. KODAMA Agent Architecture

Communication

From the viewpoints of Kodama system, the communication cost is one of the most important performance factors. Agents will register themselves and their abilities to one another at the beginning of or during

the execution. Therefore, one of the major features of Kodama is that, agents can be added to or removed from the application at runtime. Each agent, upon receiving input with an "Is-This-Yours?" performative, attempts to interpret the input by itself. If the interpretation is successfully done, the agent will report success using the "It-Is-Mine" performative with a confidence factor that reflects the similarity between the input query and its Web page contents. On the other hand, if the agent can not interpret the query as its own, before reporting failure, the agent checks with its down-chain agents. If all down-chain agents report "Not-Mine," this agent will also report "Not-Mine" to its requesting agent. If at least one down-chain agent is able to interpret the input successfully and reports back with "It-Is-Mine," this agent will also report success. Agents receiving a "This-Is-Yours" request may reinterpret the delegated input query, or they may use pre-stored interpretation of it.

Register	Agents make each other aware of their existence.
This-Is-Yours	An agent announces another agent as responsible for handling certain input.
Is-This-Yours?	An agent that can not interpret a particular input requests interpretation from down-chain agents.
Not-Mine	Down-chain agent has failed to interpret input sent down with an <i>Is-This-Yours?</i> Performative, "confidence zero".
It-Is-Mine	Down-chain agent has been successful in interpreting input sent down with an <i>Is-This-Yours?</i> Performative.
Learn	A new interpretation policy is suggested to an agent or an agent is asked to modify the weight of an existing one.

Table 1. Kodama Inter-Agent Performative Messages

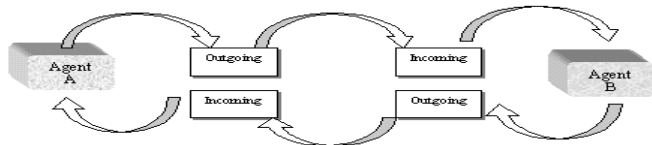


Figure 2. Message Passing and Queuing in Kodama

Kodama is a concurrent system in the architecture level and in the implementation level. Each agent has a message queue and looks it while communicating with its down-chain agents (Figure 2). Kodama agents communicate through message passing using predefined general performatives (Table 1). Agents distributed over a collection of hosts must send problem parameters and results to other agents. Among distributed agents on a network, the information is typically communicated with explicit message-passing calls. Kodama agent passes a communication message with its down-chain agents using the best-first technique.

Web Site Agentification

Cooperating intelligent Kodama agents are employed to agentify the Web where the hyper structure is preexisting in the form of Web links (Figure 3) [William *et al.*, 2000]. Our system uses three types of Kodama agents in the agentification mechanism (Figure 4) for searching the Web. A Server Agent (SA) assigned to each Web server, a Web Page Agent (WPA) assigned to each Web page, and a User Interface Agent (UIA) assigned to each user [Helmy *et al.*, 2000a]. There is a clear mapping between the problem of searching the Web in Kodama system and the classic AI search paradigm. Each WPA of the agentified Web pages is a node, and the hypertext links to the other down chain WPAs are the edges of the graph to be searched. In typical AI domains a good heuristic will rate nodes higher as we progress towards some goal node [Youngblood, 1999]. In the Kodama system domain, the heuristic models how a page is relevant to the given query.

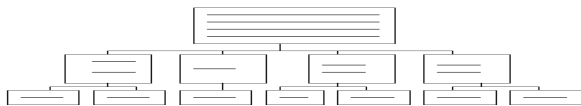


Figure 3. A Schematic view of Linked Web Pages

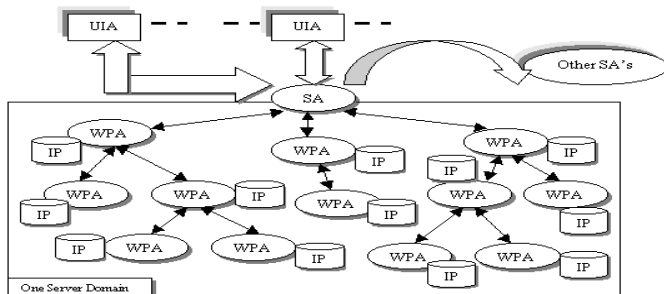


Figure 4. The Hierarchical of an Agentified Domain

A best-first search algorithm is used by the SA and the WPAs while interacting with the down chain agents. It has been slightly modified so that it will finish after reaching a predefined depth value, and return the best WPAs, which have a good relevancy to the given query.

Server Agent

A SA is assigned to one Web server to be responsible. The SA starts from the portal address of the Web server and creates the hyper structure of WPAs based on the hyper link structure in the Web server. The SA knows all WPAs in the server and works as a gateway when a WPA communicates with each other or with one in another server. The SA initiates all WPAs in its server when it starts searching relevant information to the user's query.

Web Page Agent

At the initialization phase, each WPA analyzes the content of its Web page. Each WPA starts with the base address when the WPA has got it from the SA with a serial ID number. Each WPA has its own parser, to which the WPA

passes a URL, and a private Interpretation Policy (IP), in which the WPA keeps all the policy keywords, found in its URL. All URLs found in its page are passed to the shared static data member, from it all other down-chain WPAs take their next URLs. Every time a WPA finishes, it registers itself to the SA and writes all the words into an IP. The WPA takes essential properties and principles by the SA to create the IP as an ontology that represents the context of the Web page as follows. The WPA sets the URL of that Web page as its name, loads the HTML document of that Web page, parses the HTML document, and extracts links, images, text, headers, applets, and title. Then the WPA eliminates the noisy words (non-informative words), stemming a plural noun to its single form and inflexed verb to its infinitive form. After that, the WPA creates its IP using an additional heuristics, in which additional weights are given to words in the title and headings of the Web page. Then, the created WPAs are registered to SA. WPA uses its IP in order to calculate the similarity between the user's query and its page. At the retrieval phase, WPAs, when received a user's query from SA initiate search by interpreting the query and/or either asking 'Is this yours?' or announcing 'This is yours,' to its down-chain WPAs. The selected WPAs and/or their down-chain WPAs of each Web server, in turn, interpret the query according to their IPs and reply the answer 'This is mine' with some confidence or 'Not mine' (0 confidence).

User Interface Agent

An UIA is implemented as a browser independent Java application. Monitoring the user-browsing behavior is accomplished via a proxy server that allows the UIA to inspect HTTP requests from its browser. Each UIA resides in the user's machine and communicates with WPAs via an SA to retrieve information relevant to the user's query, and shows the results returned by the WPAs to the user after filtering and re-ranking them. It receives user's responses of his/her interest (or not interest) to the results and regards them as rewards to the results. The UIAs in Kodama system look over the shoulders of the users, record every action into the query history file. After enough data has been accumulated, the system uses this data to predict a user's action based on the similarity of the current query to already encountered data.

Relevancy Algorithm with User's Query History and Bookmark Files (UP) by UIA

Recording and analyzing user's accessing history and bookmark by the UIA are quite important to catch his/her preferences. The query history file, contains information about previously visited pages for specific queries, and the bookmark file, contains a user's hot-list of Web links, will be scanned by the UIA at first to find relevant answers to the given query. A query history file (Figure 5) records the URL that a user selected, the number of occurrences that

this URL is visited, both time of visiting and leaving, and the query. The bookmark file (Figure 6) records the URL, the number of occurrences that this URL is visited, bookmarking time of the URL, and its title. The query and the title fields in query history and bookmark files are represented as a vector of keywords sorted in alphabetical order, a weight value is assigned to each keyword to reflect the correlation between the keyword and the URL and is modified according to the User's Responses (\mathfrak{R}).

URL	No. of Visiting	Time of Visiting	Time of Leaving	Query		
URL ₁	N ₁	T ₁	To ₁	k ₁₁ ,w ₁₁	...	k _{1m} ,w _{1m}
...
URL _x	N _x	T _x	To _x	k _{x1} ,w _{x1}	...	k _{xm} ,w _{xm}
URL _n	N _n	T _n	To _n	k _{n1} ,w _{n1}	...	k _{nm} ,w _{nm}

Figure 5. User's Query History File Representation

URL	No. of Visiting	Time of Bookmarking	Title		
URL ₁	N ₁	T ₁	k ₁₁ ,w ₁₁	...	k _{1m} ,w _{1m}
...
URL _x	N _x	T _x	k _{x1} ,w _{x1}	...	k _{xm} ,w _{xm}
URL _n	N _n	T _n	k _{n1} ,w _{n1}	...	k _{nm} ,w _{nm}

Figure 6. User's Bookmark File Representation

User's Responses (\mathfrak{R}) are *Useless, Not very useful, Mildly interesting, Neutral, Interesting and Bookmark*. Each of these responses has a value between 0 and 1. When looking up relevant URL from the UP, the UIA calculates similarities as follows:

First: We define equations to calculate the similarity between a user's query and his/her query history file. Assume we have a query history file or a bookmark file of n URL lines gathered. $Q_{in} = \langle k_1, k_2, \dots, k_n \rangle$ stands for a vector of keywords sorted in alphabetical order, of the query given by the user. $Q_j = \langle K_{j,1}^h, K_{j,2}^h, \dots, K_{j,m}^h \rangle$, ($1 \leq j \leq n$) stands for the vector, sorted in alphabetical order, of the query of j th line in the user's query history file, where $K_{j,i}^h = k_{j,i}^h \cdot w_{j,i}^h$, $k_{j,i}^h$ is the i th keyword in the j th line and $0 \leq w_{j,i}^h \leq 1$ is its weight.

Similarly, $T_j = \langle K_{j,1}^b, K_{j,2}^b, \dots, K_{j,l}^b \rangle$ and $K_{j,i}^b = k_{j,i}^b \cdot w_{j,i}^b$ are defined for the title of j th line in the user's bookmark file. The weight $w_{j,i}^h$ and $w_{j,i}^b$ are incrementally computed with the number t_j of visiting to URL_j .

$$w_{j,i}(t_j + 1) = \rho \cdot w_{j,i}(t_j) + (1 - \rho) \cdot \mathfrak{R} \quad (1)$$

Where $w_{j,i}$ means $w_{j,i}^h$ or $w_{j,i}^b$, and $0 \leq \mathfrak{R} \leq 1$ is a user's response described above. Initial value $w_{j,i}(1)$ is set by the user's first response. $0 < \rho \leq 1$ is a weight of how much the user's response history should be accumulated. Notice that $w_{j,i}$ means the accumulated user's preference of keyword in the j th line. ρ is a function of t_j , i.e.,

$\rho(t_j)$, and $\rho(t_j)$ depends on how long user's response history upon the keyword will be involved in calculating and adapting the next weight $w_{j,i}(t_j + 1)$.

For instances $\rho(t_j) = 1/2$ has the effect of weighting $(1/2)^t$ to the t times past response. $\rho = \frac{t_j}{t_j + 1}$ means to keep track of all the user's history, and $\rho = 0$ means to discard the user's history while calculating the weight value. The value of ρ reflects how much the system trusts the user's current response. One way to automate this heuristic is to calculate for instance the variance of user's past responses (\mathfrak{R}) and predicts the value of ρ .

Now, we calculate the similarity S_j^h between Q_{in} and the query field of j th line of the user's query history file, and similarity S_j^b between Q_{in} and the title field of j th line of the user's bookmark file.

$$S_j^h = \sum_i w_{j,i} \cdot g(k_i) \quad (2)$$

$$S_j^b = \sum_i w_{j,i} \cdot g'(k_i) \quad (3)$$

Where, $g(k_i) = 1$ if $k_i \in Q_{in} \cap Q_j$, otherwise $g(k_i) = 0$, and $g'(k_i) = 1$ if $k_i \in Q_{in} \cap T_j$, otherwise $g'(k_i) = 0$.

Also, we calculate the similarity S_j^{url} between Q_{in} and the URL of j th line.
$$S_j^{url} = \frac{s_{url}}{c_{in} + d_j - s_{url}} \quad (4)$$

Where, $c_{in} = |Q_{in}|$, $s_{url} = |Q_{in} \cap URL_j|$, $d_j = |URL_j|$, and URL_j stands for the set of words in the URL of j th line. Then, the total similarity between user's query and his/her query history file is calculated by using equation (5), with a heuristic-weighting factor $0 \leq \alpha \leq 1$.

$$\arg \max_j \left(\alpha \cdot S_j^{url} + (1 - \alpha) \cdot S_j^h \right) \quad (5)$$

Second: By the same way as the *first* step, we calculate the total similarity between the user's query and his/her bookmark file, using a heuristic-weighting factor $0 \leq \beta \leq 1$:

$$\arg \max_j \left(\beta \cdot S_j^{url} + (1 - \beta) \cdot S_j^b \right) \quad (6)$$

IP Representation and Relevancy by WPA

An IP is used by the WPA to decide whether or not the keywords in the query belong to the WPA. The IP is a vector of important terms, which is extracted and weighted by analyzing the contents of the Web page. Since the terms are not all equally important for content representation of IP vector of each WPA, an importance

factor (λ) is assigned to each term and decided by the kind of HTML tag, in which the term is included in the Web page. This means that WPA will emphasize/de-emphasize some keywords based on the value of λ . The WPA calculates the weight of the term and constructs its IP vector from the number of appearance (tf) and the kind of tag, which includes the term within the Web page (e.g., in title, in header, in link, is bold, underline, italic, keyword or normal), using equation (7).

$$w_{ik} = \lambda_k \cdot tf_{ik} \quad (7)$$

Where w_{ik} stands for the weight of term i in position k , and tf_{ik} stands for the number of occurrences that term i appears in position k specified by HTML tags. λ_k stands for the weight decided by the kind of HTML tag k that includes the term i in the Web page. The total weight of a term i in the IP is the sum of all the weights in the HTML document of the Web page and is calculated by:

$$w_i = \sum_{k=1}^n w_{ik} \quad (8)$$

Where n is the number of HTML tags within the Web page. The WPA_i calculates the confidence factor that reflects the relevancy between the input query vector Q_{in} and its IP vector using equation (9).

$$Sim(Q_{in}, IP) = \frac{\sum_{i=1}^n h(k_i) \cdot w_i}{\sqrt{\sum_{i=1}^n h(k_i)^2} \cdot \sqrt{\sum_{i=1}^n w_i^2}} \quad (9)$$

Where, $h(k_i) = 1$ if $k_i \in Q_{in} \wedge k_i \in IP$, otherwise $h(k_i) = 0$.

Query and URL Similarity

The WPAs calculate the similarity between the user's query and their documents based on the terms they have in both of the IPs and the hyperlink structures of the WPAs. This similarity function is based on both query-IP and query-URL similarities. It is a hybrid similarity function that includes two components. The S_{ij}^{Q-link} component measures the similarity between the user's query i and the URL of a Web page j , and is calculated using equation (4). The S_{ij}^{Q-IP} component measures the similarity between the user's query i and the IP of a WPA of Web page j , and is calculated using equation (9). The whole similarity is calculated using equation (10) and $0 \leq \delta \leq 1$.

$$S_{ij}^{Total} = (\delta \cdot S_{ij}^{Q-Link} + (1 - \delta) S_{ij}^{Q-IP}) \quad (10)$$

Exploiting User's Preferences by UIA

The UIA takes care of the user's interests and preferences to assist of finding relevant Web pages his/her query. The

user can provide positive or negative feedback to the retrieved document. In contrast to other systems that learn a UP and use it to determine relevant documents [Pann and Sycara, 1996], [Chen and Sycara, 1998], Kodama's UPs are continuously evolving according to the dynamically changing user's preferences. The followings are the UIA's job stream (Figure 7):

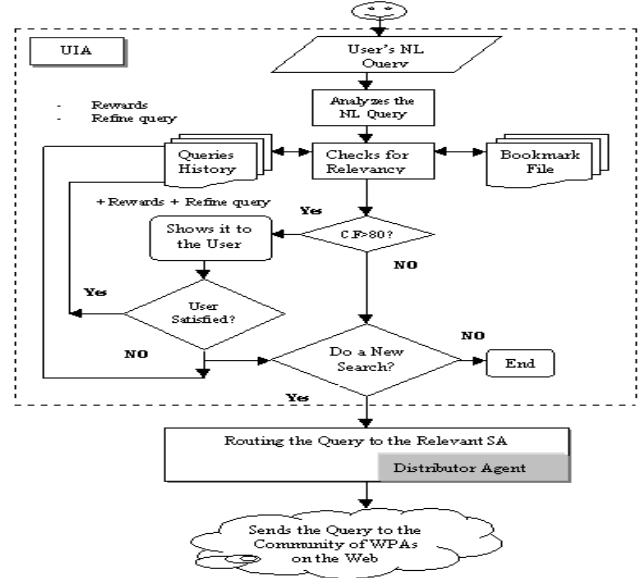


Figure 7. A Schematic Diagram of UIA's Behaviors

- (1) The user starts by sending a Natural Language (NL) query to the UIA.
- (2) UIA analyzes the NL query using a simple NL algorithm, and transforms it to Q_{in} .
- (3) UIA calculates the similarity with the method described above and looks for relevant URLs in UP by using equations (5) and (6).
- (4) If UIA finds relevant URLs then shows them and asks the user whether the user is satisfied or wants to search the Web.
- (5) If UIA could not find in its UP files any URLs relevant to Q_{in} then UIA routes Q_{in} to a relevant SA, which in turn forwards it to its community of WPAs.
- (6) The UIA receives the search results returned by the WPAs via the SA. The results consist of a set of contents of Web pages.
- (7) The UIA takes a set of queries, whose similarity to Q_{in} is over the predefined threshold value, from the UP to expand Q_{in} . Then, the UIA makes a vector from them and Q_{in} to be used in the filtration process.
- (8) The user explicitly marks the relevant documents using UIA's feedback. This response is stored in the response field of the UP.

The most relevant SA to the user's query selected either by having the user explicitly define the SA as a specific portal or by having the system determines by itself by examining the user's bookmark, query history and SA's attributes.

Learning and Adaptation in KODAMA

Because we are automatically creating the IP of each WPA based on the contents of its corresponding Web page and creating the UP based on the user's preferences, it is necessary to improve the IP and UP dynamically. There are several approaches that can be used to learn a UP [Budzik and Hammond, 1999], [Joachims *et al.*, 1997]. In Kodama a WPA interacts with the UIA as following (Figure 8). The UIA sends the Q_{in} to WPAs through the SA. The WPAs choose an action to perform based on Q_{in} and send the results back to the SA, which in turn forwards the results to the UIA. The UIA presents the results to the user. The user can click on those URLs and the UIA opens its browser to that specific URL. The user can check and evaluate this URL's contents and sends feedback through UIA's menu.

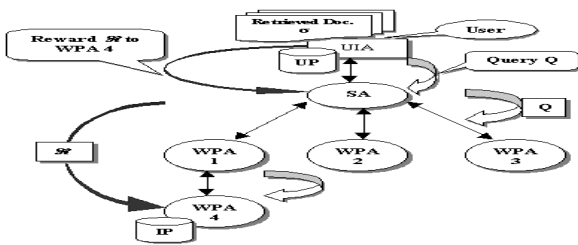


Figure 8. Interaction between WPA & UIA

Adaptation in WPA

The WPA allows only relatively small change of the term's weight based on the user's response, because adding/removing some terms into/from the IP may change the context of the Web page. When the user says the page is interesting, WPA changes the weight of the terms in its IP, if and only if these terms appear in Q_{in} , in order to make better match with the query to be entered next time. This means that, WPA will emphasize/de-emphasize some terms, frequently, reflecting the user's responses. If the user's query is $Q_{in}=\{q_1, q_2, \dots, q_n\}$, then the WPA checks if $q_i \in IP$ then changes its weight w_i by adding a reward's value \mathfrak{R}_i to be $w_i + \mathfrak{R}_i$, else ignores it.

Adaptation in UIA

The UIA picks a number of keywords from the title and the headers of the selected document K_s in addition to the keywords of Q_{in} and creates a new list of keywords for the feedback K_f . Where, $K_f = Q_{in} \cap K_s$. According to \mathfrak{R} , the UIA will do the followings on the UP files:

- ◆ Modify the weight of the keyword using equation (1).

- ◆ Modify the number of visiting.
- ◆ If one of the keywords does not exist in the query field then adds it with an initial weight reflecting the user's response.
- ◆ Refine the contents of the UP files by deleting the keywords that have weights less than a predefined threshold value.
- ◆ If the selected URL does not exist in the UP then adds a new record and initializes its query field.

This process iterates until the user's is satisfied with the retrieved information. By this way, the UP will evolve over time to reflect the user's interests. Also, the keywords of the query and title fields continually moved closer to or away from their URLs.

Experimental Results

We have performed several experiments to make a consistent evaluation of Kodama system performance. The results we have obtained for users, who used the system from 10th of October until 25th of December, verify the facts that Kodama can learn and adapt to the UP over time. Also, the idea of Web page agentification promises to achieve more relevant information to the user.

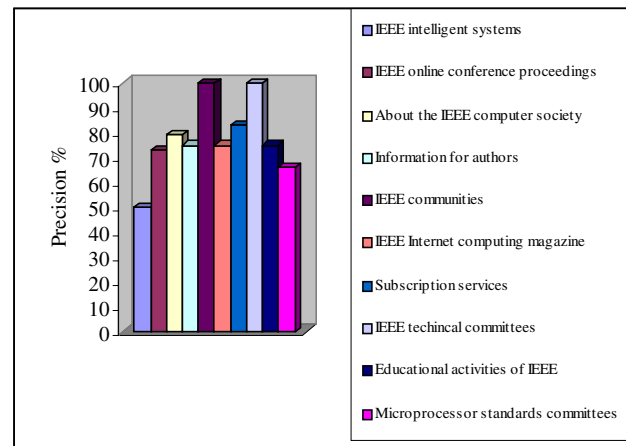


Figure 9 Precision of the Retrieved URLs to the User's Queries to the IEEE Agentified domain

In the first experiments, we attempted to verify that the mechanism of agentifying the Web is useful for retrieving relevant information. We agentified several Web servers by giving the portal address of the Web servers to the system, the system creates the hyper structure of the WPA communities based on the hyperlink structure of each Web server. Then, we calculated the Precision of the retrieved URLs to user's queries. Figure 9 shows the Precision for user's queries to the IEEE agentified domain <http://computer.org/>. The number of agentified Web pages in this Web server is about 2000. Also, we agentified the AAAI domain, <http://www.aai.org/>. The number of agentified Web pages in this Web server is 2736. Figure

10 shows the Precision for user's queries to the AAAI's agentified domain. The results depicted in Figures 9 and 10 show that the idea of Web page agentification promises to achieve relevant information to the users and also, promoted using Kodama as a PinPoint IR system.

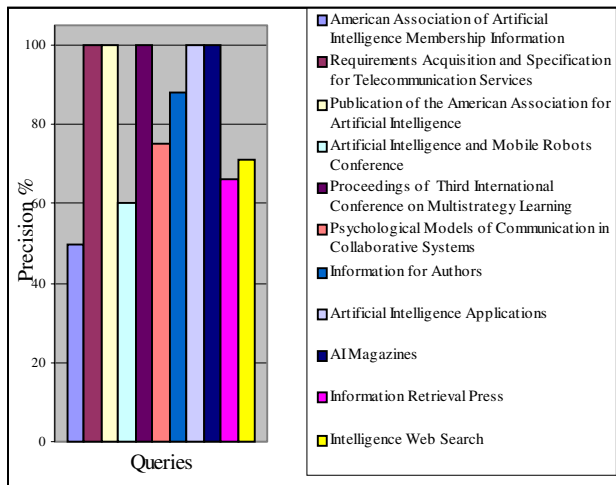


Figure 10 Precision of the Retrieved URLs to the User's Queries to the AAAI Agentified domain

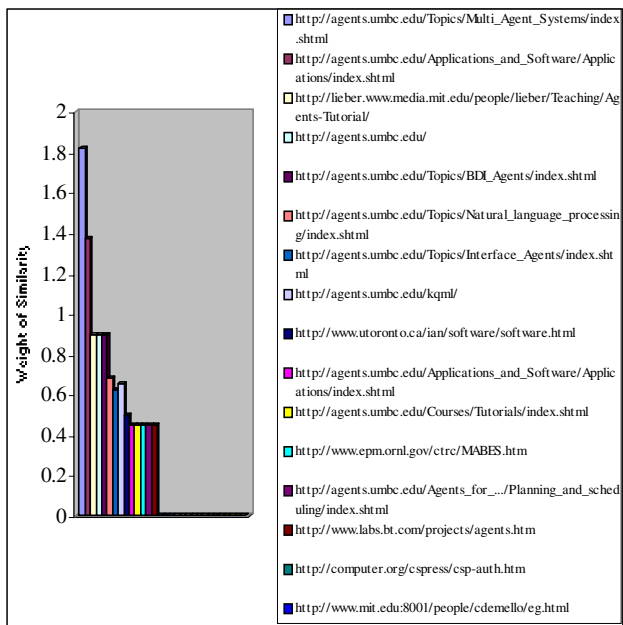


Figure 11. URL's Correlation with the Relevant Keywords

In the second experiments, we measured how well Kodama can get correlation between the contents of the URLs and the queries over time to predict the UP. The user starts by giving the following five queries three times, “Conferences and workshops of agents,” “Natural language processing agent,” “Electronic commerce agent systems,” “KQML agent communication language,” and “Collaborative intelligent interface agents.” At this point, the system has already been

customized to user's current interest and the URLs get correlated with the queries by inserting new keywords, deleting non-relevant keywords and modifying the weights of the existing keywords. After that, the user gives more general and ambiguous query, “*intelligent agent systems.*” The highest-weighted URLs, which were retrieved and satisfied the user, are the most relevant URLs to the given query in the UP. Figure 11 shows the URLs in descending order of similarity within the UP.

In the third experiments, we measured how well Kodama is being able to adapt to the UP over time and to get a good correlation between each URL and its relevant keywords. In order to understand the experiment, we define a Fitness value, which will show the correlation between the weights of keywords calculated by UIA and user's actual interest to each keyword, as follows.

(1) User's actual interest: $S_j = \sum_{k=1}^m b_k \cdot W_k$, where W_k is the weight of keyword_k, and $b_k = 1$ if the user judges keyword_k in the URL_j as relevant for his/her query, else $b_k = 0$.

(2) Interest calculated by UIA: $T_j = \sum_{k=1}^m W_k$.

We define the Fitness value $F_j = S_j / T_j$, which reflects the correlation between the two interests for URL j.

In the experiment, a user gave fifteen different queries, each of which consists of 1 to 5 keywords, then, after frequent interactions of retrieval, the user checked the relevancy of each keyword in the retrieved URL, then Fitness value was calculated for each URL in the UP. The Fitness values calculated after five and ten times retrieval interactions are shown in Figure 12. Figure 12 shows that the values of S and T are converging over time to each other, and this means that UIA is being able to predict and adapt to its actual user's interests.

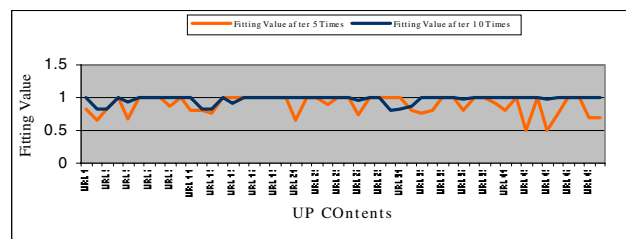


Figure 12 Converging to the User's Interest over Time

In the fourth experiments, we measured how well the feedback mechanism of UIA enables users to access to more relevant information with high relevancy. We define the Feasiblensness of the feedback mechanism by M/N . Where N means the number of queries given by a user, and M means the number of the retrieved results at highest rank with which the user get satisfied. In the experiment, a user initially starts by giving a set of

ambiguous and non-sense queries. At this point, URLs are retrieved and held in UP. Then, the system repeats the interaction process, in which the user gives a query, gives back the evaluation to the retrieved URLs, and the rank of URLs in UP is changed according to the user's responses. In the experiment, the interactions were repeated ten times. The Result of the experiment is shown in Figure 13. Figure 13 shows that the Feasibleness gradually goes up with time, and this means that UIA is helpful for users to retrieve relevant URLs.

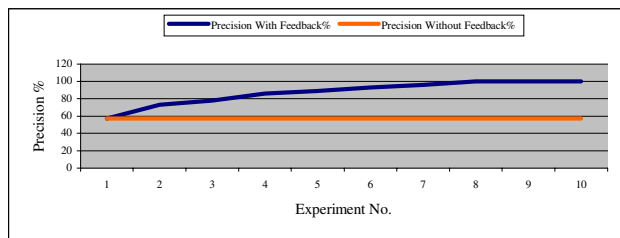


Figure 13 Feasibleness while Catching the User's Behavior

Conclusion and Future Work

This paper discussed a multi-agent-based approach to build scalable information searching techniques to let users retrieve information in highly distributed and decentralized databases, such as the Web. In this sense, the multi-agent system is not composed of a few agents, but rather composed of possibly large number of agents, which collectively try to satisfy the user's request. We reported methods to agentify the Web, and to exploit UP & IP adaptively on the Kodama system. We carried out several experiments to investigate the performance of the Kodama system. Through these experiments, we ensure that the idea of Web page agentification promises to achieve relevant information to the user. So, Kodama can be used as a PinPoint IR system that learns and adapts to the user's preferences over time. The system is able to change the weights of some keywords and classifies URLs in query history and bookmark files in a way that reflects user's interest in these keywords of the related URLs. Future step in Kodama is extending our experiments in multiple SA domains and developing a smart query routing mechanism in the UIA for routing the user's query. Routing refers to the process of selecting the SA to be queried and forwarding queries to it. UIA will route the query to an appropriate SA, instead of sending the query to all SAs and gathering a large amount of noisy Web pages. In such a situation, the UIA and SA need an intelligent query routing mechanism that suggests the most relevant SA based on user's query history and some attributes of the SAs. By sending the queries to the relevant SAs, the traffic of the network will be minimized, and also users will receive only a relevant information.

References

[Budzik and Hammond, 1999] Budzik J. and Hammond K. "Watson: Anticipating and Contextualizing Information Needs", in Proceedings of Sixty-second annual Meeting of the American Society for Information Science.

[Chakrabarti *et al.*, 1998] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan, "Automatic Resource Compilation by analyzing Hyperlink Structure and Associated Text", Proceedings of the 7th WWW Conference.

[Chen and Sycara, 1998] Liren Chen and Katia Sycara, WebMate: A Personal Agent for Browsing and Searching", Proceedings of the Second International Conference of Autonomous Agents, Minneapolis/ST, MN USA, May 9-13, 1998, pp.132-138.

[Edmund *et al.*, 2000] Edmund S. Yu, Ping C. Koo, and Elizabeth D. Liddy: Evolving Intelligent Text-based Agents, Proceedings of the 4th International Conference of Autonomous Agents, June 3-7- 2000, Barcelona, Spain, pp.388-395.

[William *et al.*, 2000] G. William, S. Lawrence and C. Giles, "Efficient Identification of Web Communities", ACM Proceedings of KDD 2000, Boston, MA, USA.

[Helmy *et al.*, 1999a] T. Helmy, B. Hodjat and M. Amamiya, " Multi-Agent Based Approach for Information Retrieval in the WWW", Proceedings of the First Asia-Pacific International Conference on Intelligent Agent Technology (IAT'99), Hong Kong, 15-17/12, 1999, pp. 306-316.

[Helmy *et al.*, 2000b] T. Helmy, T. Mine, G. Zhong, M. Amamiya, "A Novel Multi-Agent KODAMA Coordination for On-line Searching and Browsing the Web", Proceedings of The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, 10-12/4, 2000, Manchester, UK, pp. 335-338.

[Helmy *et al.*, 2000c] T. Helmy, T. Mine, G. Zhong and M. Amamiya, " Open Distributed Autonomous Multi-Agent Coordination on the Web", Proceedings of The Seventh International Conference on Parallel and Distributed Systems Workshops, pp. 461-466: July 4-7, 2000, Iwate, Japan.

[Helmy *et al.*, 2000d] T. Helmy, T. Mine and M. Amamiya, "Adaptive exploiting User Profile and Interpretation Policy for Searching and Browsing the Web on KODAMA System", Proceedings of the 2nd International Workshop on Natural Language and Information Systems NLIS, London, Greenwich, United Kingdom, September 4-8, 2000, pp. 120-124.

[Hodjat and Amamiya, 2000] B. Hodjat and M. Amamiya, "Applying the Adaptive Agent Oriented Software Architecture to the Parsing of Context Sensitive Grammars", IEICE TRANS. INF. & SYST., VOL. E83-D, No.5 May 2000.

[Kleinberg, 1999] J. Kleinberg "Authoritative sources in a hyperlinked environment", ACM Journal, 46(s), PP. 604-632.

[Menczer and Monge, 1999] F. Menczer, A.E. Monge:" Scalable Web Search by Adaptive Online Agents: An InfoSpiders Case Study". Intelligent Information Agents.

[Pann and Sycara, 1996] K. Pann, A. And Sycara, K. " A Personal Text Filtering Agent", Proceedings of the AAAI Stanford Spring Symposium on Machine Learning and Information Access, Stanford, CA, March 25-27, 1996.

[Youngblood, 1999], G. Michael Youngblood, "Web Hunting: Design of a Simple Intelligent Web Search Agent", ACM Crossroads Student Magazine.