

Adaptive User Interface of a Web Search Engine by Organizing Page Information Agents

Seiji Yamada and Fumihiko Murase
CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatsuta, Midori, Yokohama 226-8502, JAPAN
yamada@ymd.dis.titech.ac.jp

Abstract: In this paper, we develop an organization method of page information agents for adaptive interface between a user and a Web search engine. Though a Web search engine indicates a hit list of Web pages to user's query, it includes many useless ones. Thus a user has to select useful pages with page information indicated on the hit list, and actually fetch the page for investigating the relevance. Unfortunately the page information is insufficient for a user, and which information is adequate depends on a user and a task. Hence we propose adaptive interface AOAI in which different page information agents are organized with user's evaluation. As results, different organizations are achieved depending on a user and a task.

1 Introduction

Accessible information through the Internet is increasing explosively as the WWW becomes widespread. In this situation, the WWW is very useful for a user who wants to search and gather information. However there is a significant issue that a user does not know relevant URLs. The practical solution is to use a search engine. A search engine provides a list of relevant Web pages (called a *hit list*) to queries from a user. Unfortunately, since adequate filtering is very hard, many irrelevant Web pages are indicated in a hit list.

Hence a user needs to select relevant pages from a hit list using information indicated on each page, and actually fetch the Web pages to verify the relevance. We call the information on Web pages in a hit list *page information*. Unfortunately the page information like a title, the size and head sentences is fixed and neither sufficient nor necessary for a user to evaluate the pages. For example, Fig.1 shows page information in a hit list indicated by *InfoSeek*, one of major search engines, and the page information is significantly inadequate because it lacks important elements like image, not-found, network traffic and so on. However adequate necessary page information depends on a user and a task, hence we hardly design interface with adequate page information in advance. Thus we propose adaptive interface in which different page information agents are organized through human-computer interaction and suitable interface is gradually constructed. The page information agents indicating different information are prepared at first. A user evaluates them through search, and they are organized. As results, adaptive interface (Norcio & Stanley 89) is achieved depending on information utilized by a user and a task. We call this framework AOAI (Agent Organization-based Adaptive Interface).

Pnadiit and Kalbag developed an assistant system for operating text on a PC, in which multiple recognition agents are integrated (Pandit & Kalbag 97). Each agent is able to extract different information on URL, phone number, place name from text in a clipboard. When plural agents obtain information and try to indicate them, they are organized into a multi-layered menu. Unfortunately the way to organize them is fixed and the system is not adaptive to user's preference.

1. [Tokyo Institute of Technology Library \(TITECH--Japan\)](#)

Pages in English and Japanese.

Relevance: 100% Date: 8 Apr 1999, Size 1.5K, http://www.libra.titech.ac.jp/welcome_e.html


[Find similar pages](#) |  [More results from www.libra.titech.ac.jp](#) | [Translate this page](#)

Figure 1 Information on a hit Web page by *InfoSeek*

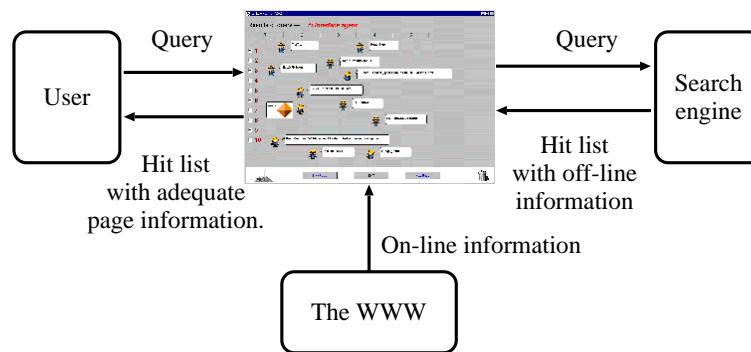


Figure 2 Overview of AOAI.

SATELIT-Agent (Akoulchina & Ganascia 97) distinguishes the expert's profile and offer a special interface to them. It can navigate a user by inferring an analogue of the user's search requirements. Also Maglio and Barrett observed users searching for information in the WWW, and build two personal Web agents (Maglio & Barrett 97). The shortcut agent can identify the repeated search patterns and suggest them for new searches, and the waypoint agent can identify and display the nodes that go together in the history. However these studies could not deal with page information in a Web search engine and not organize interface agents.

2 Overview of AOAI

AOAI is adaptive interface between a Web search engine and a user like shown in Fig.2. AOAI receives queries from a user and gives it to a search engine as it is. Then it obtains the hit list to the queries from a search engine and indicates them to a user with useful information for selecting a page which he/she wants. Note that AOAI deals with the two kinds of page information: off-line and on-line information. The *off-line information* is obtained from a hit list of a search engine, and *on-line information* is obtained from a Web page which was actually fetched on-line. In contrast that the off-line information is quickly indicated and old, the on-line information is slowly indicated and recent. The page information is indicated by each PIA (Page Information Agent) in an IUW (User Interface Window) of AOAI. The followings show a main procedure of AOAI.

1. Giving a query to AOAI.
2. A hit list is indicated on the IUW. Fig.3 shows the initial IUW. When a user points a *target page number* ((A) in Fig.3) which he/she wants by a mouse cursor, all the PIAs indicate page information on the page.
3. Select the Web page which he/she wants with page information from PIAs.
4. Double click the target page number and see it through a Web browser.
5. If the page is relevant, a user evaluates the PIAs whose information contributed to the selection, and organization is done. Otherwise, go to 3 and a user selects other pages. The positive/negative evaluation is done by clicking a right/center button of a mouse on a PIA.
6. This query is finish by pushing "Exit" button ((D) in Fig.3) , and if necessary, AOAI organizes PIAs using the procedure described later. Go to 1 with next query.

A user evaluates PIAs typically when he/she obtained relevant pages. However AOAI allows a user to evaluate them anytime. Thus this evaluation hardly gives cognitive load to a user. Furthermore, for seeing page information effectively, a user can directly manipulate PIAs by drag and drop it anytime, and arrange them as he/she likes. We assume that AOAI should integrate agents which were placed closely by user's direct manipulation.

Additionally to the above procedure, a user is able to deconstruct an integrated PIA by drag&drop it on the factory icon ((B) in Fig.3). Also after embedding, a user can decompose embedded agents by clicking a trash box icon.

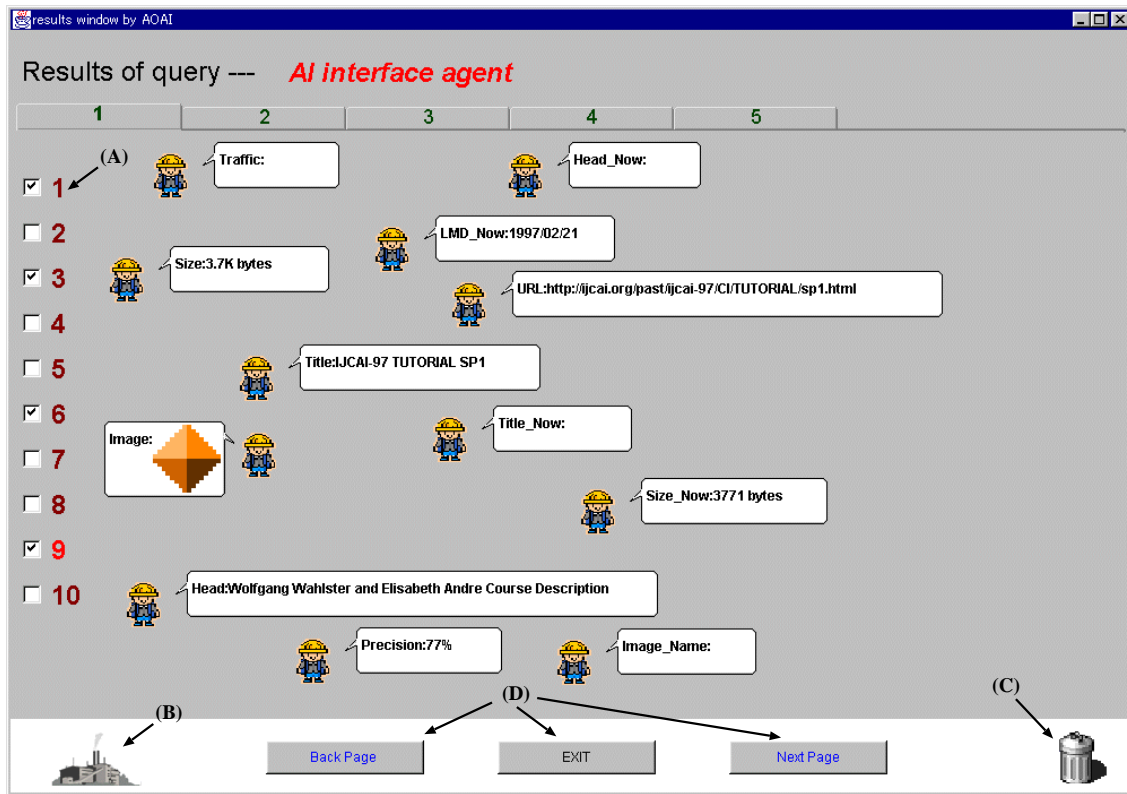


Figure 3 Interface window (initial state).

3 Page Information Agents

AOAI provide 12 PIAs like the followings. Every agent is designed and implemented in advance. An agent has human-like appearance which changes depending on the growth value mentioned later and a balloon in which a message from the agent to a user is indicated. Some agents indicate the off-line information and other agents indicate on-line information with “_Now”. A user can utilize any of them.

- **URL agent:** This shows the URL of a target page obtained off-line.
- **Traffic agent:** This agent investigates the traffic on network connection to a target page on-line. It shows “Server down”, “Light”, “Heavy” depending on the response.
- **Title agent:** This shows the title of a target page obtained off-line.
- **Title_Now agent:** This shows the title of a target page obtained on-line.
- **Head agent:** This stands for the first sentence of a target page off-line.
- **Head_Now agent:** This stands for the first sentence of a target page obtained on-line.
- **Precision agent:** This shows the rate of precision obtained off-line.
- **Size agent:** This agent investigates the size of a target page off-line.
- **Size_Now agent:** This agent investigates the size of a target page on-line.
- **LMD agent:** This agent indicates the last modified date obtained off-line.
- **LMD_Now agent:** This agent indicates the last modified date of a target page using a similar on-line way to a file-size agent.
- **Image_Name agent:** This shows the file name of the image investigated on-line.
- **Image agent:** This directly indicates an image in a Web page.

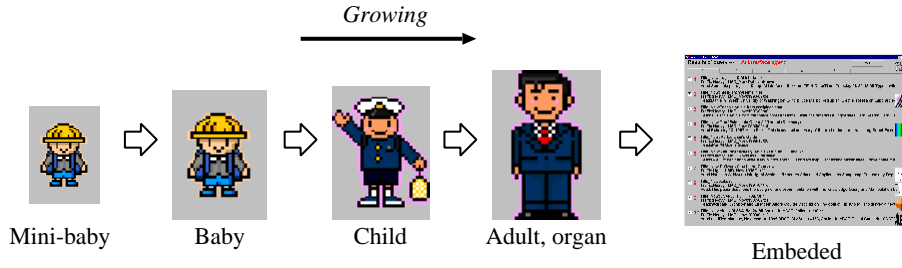


Figure 4 Growing character for each stage.

4 Organizing Page Information Agents

4.1 Agent Properties

A PIA A_i has the following properties on organization and itself activity.

- **Growth G_i :** This property indicates A_i 's activity of which is increased/decreased by human evaluation. A method to compute G_i will be described in the next subsection. This property is used for growing of a PIA.
- **Stage S_i :** This takes seven values indicating A_i 's stage: *trash*, *mini-baby*, *baby*, *child*, *adult*, *organ* and *embedded* depending on the growth G_i . The appearances are shown in Fig.4. An agent grows from baby to adult, and finally is embedded into a hit list, where an agent disappears and only its page information is indicated. A PIA in a trash stage is moved into the trash box icon ((C) in Fig.3).
- **Page information I_i :** This is a list of page information which A_i indicates.
- **Position $P_i(x, y)$:** Position (x, y) of an agent A_i in a window is used for computing the distance D_{ij} between an agent A_i and an agent A_j .

4.2 Human Evaluation and Growing Page Information Agents

In AOAI, PIAs are organized by human evaluation. The growth G_i is updated by a user's click using the following formula. Since this G_i is computed temporally within last 10 searches, not cumulatively, AOAI is adaptive to the change of the domain in which a user wants information, and this is experimentally verified in section 5.2.

$$G_i = (\text{No. of positive eval. in last 10 searches.}) - (\text{No. of negative eval. in last 10 searches.})$$

We define *growth threshold* ψ_G and *decline threshold* ψ_E for controlling the growth of PIAs. When the G_i becomes more/less than ψ_G/ψ_E , PIA A_i grows/declines into the upper/down stage with resetting $G_i = 0$. However the embedding needs a special procedure mentioned later.

In addition to integration, AOAI provides a way to eliminate useless PIAs. It eliminates a *mini-baby* agent A_i with G_i is less than an ψ_E into the trash box. As results, useful PIAs survive and useless ones disappear in a UIW.

4.3 Organization Procedure

The *relation* R_{ij} between two PIAs A_i and A_j is constructed when the following precondition is satisfied. This condition means that the two *organ* agents are closed within ψ_D . As mentioned before, seeing page information effectively, a user is able to directly manipulate PIAs anytime, and arrange them as he/she likes. Thus we assume that AOAI should integrate agents which were placed closely by user's direct manipulation. We introduce ψ_D *distance threshold* for controlling the organization. As ψ_D and ψ_G are small, agents tend to be organized quickly. The D_{ij} is a mean distance between A_i and A_j for last 10 searches.

$$(D_{ij} < \psi_D) \wedge (S_i = \text{organ}) \wedge (S_j = \text{organ})$$

We use undirected graph representation for organizing PIAs. The nodes and arcs correspond to agents and constructed relations. In the graph, groups of PIAs to be integrated are determined by investigating maximal connected sub-graphs. Note that we do not use maximal cliques because they are too restricted. Since we can apply depth-first search to extract maximal connected sub-graphs, the computational cost for extracting all the groups is $O(n)$

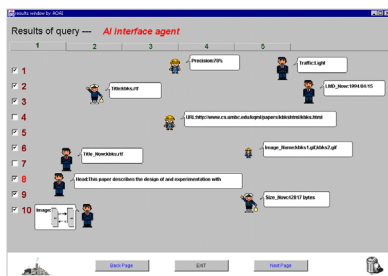


Figure 5 Growing agents.

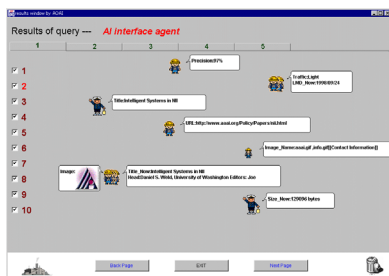


Figure 6 Agent integration.



Figure 7 Embedded agents.

where n is the number of agents, and this is sufficiently fast to AOAI. When organization is done, the new agent A_k integrated of agents $A_i \cdots A_j$ has a balloon including all of the $I_i \cdots I_j$ and starts as baby having $G_k = 0$.

As organization is progress, useful PIAs are gradually integrated into a single one. When all the useful agents are organized into a single agent A_i , it is embedded into a hit list and I_i is indicated in the list.

We have been fully implemented AOAI using JAVA. First of all, a user input a query to AOAI. Fig.3, Fig.5, Fig.6 and Fig.7 show the executed examples of AOAI with a query "AI interface agent". Fig.3 stands for an initial AOAI having all of the 12 PIAs. As seeing from the figure, all the agents are initially babies. In Fig.3, the check box in the left side of a target page number is checked when all the on-line information on the page is obtained. A user moves a cursor on a Web page number and PIAs indicate own information on it in their balloons. A user investigates information on each Web page number and double-clicks the Web page number which he/she wants to actually see the page through a browser. After repeating this process, a user finds a relevant Web page and evaluates PIAs.

As the agents get user's evaluation, they become "child" and "adult". Fig.5 shows the scene in which the five agents became "adult", two agents got "child" and two agents were already eliminated. In Fig.6, the two groups of "organ" agents satisfied the condition of organization and they are organized into two "baby" agent having their integrated page information in their balloons. Finally the single PIA made of Title, Traffic and Head agents survived and was embedded into a hit list like Fig.7.

5 Experimental Evaluation

We made three types of experiments for evaluating AOAI. For all the following experiments, the growth, decline and distance threshold were experimentally set 10, -3 and 200 pixels. Also we used *infoseek* as a search engine of AOAI for all the experiments.

5.1 Exp-1: Various Embedded Agents

If every user utilizes the same page information for finding relevant Web pages, AOAI does not need to be adaptive to a user and we can design interface using the common page information in advance. Thus we first investigate whether users use different page information in a hit list. This is examined by variation in embedded agents for each user. We made experiments in which nine subjects (master course students) freely used AOAI and embedded agents are obtained finally. Table 1 shows the results. The mean time and the mean number of searches until embedded were 165 sec. and 9, respectively. As seeing from this table, subjects embedded agents differently and no PIA embedded for all the subjects. This results supports that AOAI needs to be adaptive to users.

5.2 Exp-2: Adaptation to the Change of a Task

Next experiments are on adaptation to the change of the domain in which a user wants Web pages. We gave four subjects the four domains ("programming", "technical paper", "prize" and "image") sequentially and they searched information on the domains. We consider the change of the domain the change of a search task. When the agents were embedded in a domain, the next domain is given to a subject. As results, we observed that all the subjects embedded agents in a domain, decomposed them for adaptation to the new search task, and embedded them again. Thus a user can be adaptive through AOAI to the change of a task.

Sub.	Embedded agents	Sub.	Embedded agents
S1	URL,Tr.,Ti.,Head_Now,LMD,Im.	S6	URL,Ti.,Ti._Now,Head,Head_Now,Im.
S2	URL,Ti.,Head,Im.,LMD,Im._Name	S7	Ti.,Ti._Now,Head,Head_Now
S3	Tr.,Ti.,Ti._Now,LMD,Im.,Im._Name	S8	URL,Tr.,Ti.,Ti._Now,Head,LMD,Im.,Im._Name
S4	URL,Ti._Now,Head_Now,Precision,Im.	S9	URL,Ti.,Ti._Now,Precision,Im.
S5	Tr.,Ti._Now,Head_Now,Precision,Im.		

Table 1 Embedded agents for each subject.

Domain	Time (sec.)		# of Web pages	
	AOAI	<i>Infoseek</i>	AOAI	<i>Infoseek</i>
program	261(158)	475(339)	4.4(2.61)	13.4(8.0)
paper	219(153)	330(279)	2.6(2.2)	11.8(10.4)
prize	551(713)	546(654)	5.9(7.4)	5.5(5.8)
image	207(115)	381(640)	1.7(1.1)	9.8(14.4)
Average	298 (367)	411(509)	3.4(4.1)	9.1(10.7)

Table 2 The results for Exp-3

5.3 Exp-3: Comparison with a Search Engine

Finally we compared AOAI with embedded agents with a search engine *infoseek*. We used two subjects for each of “programming”, “technical paper”, “prize”, and three subjects for “image”, and investigated the the averages of search time and the number of Web pages seen by a subject until relevant Web page was found. The results are shown in Table 2. The bracketed number stands for standard deviation. Though the search time of AOAI is less than that of a search engine in average, the difference is not statistically significant by *t*-testing. However the difference of the number of Web pages seen by a subject is statistically significant, and AOAI outperforms a search engine over two times. Since seeing Web pages is hard cognitive load for a user, we conclude the cost of AOAI for a user is significantly less than that of a search engine.

6 Conclusion

We proposed AOAI: adaptive interface in which different page information agents are organized through man-machine interaction. In AOAI, the page information agents indicating different information on a hit list like file-size, network traffic and a page title are prepared at first. A user evaluates them through a use of a search engine, and they are organized based on the evaluation. As results, different organization is achieved depending on a user and a task. By making experiments with subjects, we found AOAI was promising adaptive interface for providing adequate page information in a hit list.

References

- [Akoulchina & Ganascia 97] Akoulchina, I. & Ganascia, J. (1997). SATELIT-Agent: An adaptive interface based on learning interface agents technology, In *Proceedings of the Sixth International Conference on User Modeling*, 21–32.
- [Maglio & Barrett 97] Maglio, P. P. & Barrett, R. (1997) How to build modeling agents to support Web searchers, In *Proceedings of the Sixth International Conference on User Modeling*, 5–16.
- [Norcio & Stanley 89] Norcio, A. F. & Stanley, J. (1989). Adaptive human-computer interfaces: A literature survey and perspective, *IEEE Transaction on Systems, Man, and Cybernetics*, 19(2), 399–408.
- [Pandit & Kalbag 97] Pandit, M. S. & Kalbag, S. (1997). The slection recognition agent: Instant access to relevant information and operations, In *Proceedings of the 1997 International Conference on Intelligent User Interfaces*, 47–52.