Agent Technology for Website Browsing and Navigation

Hsieh-Chang Tu, Michael L. Lyu and Jieh Hsiang
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
email:{tu,michael,hsiang}@csie.ntu.edu.tw

Abstract

A website *is a repository of information which can be accessed through a Web browser. There are more and more websites that are devoted to a specific topic. Such websites are attractive because they are often better organized, systematic, and are the easiest places to gather information that the user needs about a special domain. This paper discusses the issue of* website browsing and navigation*, that is, traversing within the confine of a website and collecting information. Although the problems posed by website browsing are similar to those of* Web browsing*, they are different in magnitude and solvability. We investigate problems encountered in website browsing and navigation and propose solutions based on agent technology. We investigate how a website browsing agent may utilize user profiles to achieve personalization, and propose a dedicated window approach to integrate all desirable functions. We illustrate how the idea works by describing a website browsing agent for navigating the NTU digital library.*

1 Introduction

The success of the World Wide Web (Web for short) opens a new era of the information age. Hundreds of millions of Web pages form a complicated information system from which people can extract valuable information. To help a user find useful Web pages, there are *search engines* which return a list of Web pages likely to be relevant to a user query. and there are *Internet agents* which assist people in accessing Web resources [2].

A (software) *agent* is a program which assists a user in managing information. Ideally, one would like an agent to possess some form of human trait, such as autonomy, social ability, reactivity and pro-activeness [10]. Since different users have different interests, it is also desirable for an agent to have *personalization* abilities [1]. However, there is no common agreement on exactly what a software agent is [2, 4].

In this paper, we introduce a *website browsing agent* for assisting a user in retrieving useful pages from a particular website designed for a specific domain. We remark that a *website* browsing agent is different from a *Web* browsing agent. While the latter may help a user in extracting information from the Web in general, the former is designed for a specific website. Such a software system is usually more useful because its design can be tailored to capture the characteristics of the chosen website.

We organize the remaining part of this paper as follow. In Section 2, we examine problems a user may encounter in website browsing and propose an agent-based solution. In Section 3, we demonstrate how our solution works by describing a website browsing agent designed for navigating and collecting information in the NTU digital library. Finally, we give concluding remarks about our approach and about future work in Section 4.

*Partly supported by Grant NSC 87-2213-E-002-012 of the National Science Council of the Republic of China

2 Website Browsing and Navigation

A *website* is a repository of information which can be accessed through a Web browser. The *homepage* of a website is a specific Web page that a user normally access first in order to entire the website. By specifying the URL (uniform resource locator) of the homepage, a user enters the website. Other pages in the same website can then be accessed either through hyperlinking or through search services provided by the site. The process of traversing within a website is called *website navigation.*

The main purpose of building a website is for users to visit and obtain useful information. There are more and more websites that are devoted to a single topic. Such websites are attractive because they are often better organized, systematic, and are the easiest places to gather information that the user needs about a special domain. We call the process of traversing through such a website and collecting information *website browsing and navigating* to emphasize that, even though the user may visit Web pages not belonging to the site (through hyperlinks), she is focusing on finding useful information from within the website. Although the problems posed by website browsing are similar to those of web browsing, they are different in magnitude and solvability. For instance, a frequent user may want to know if the contents of the website has been modified since the last visit and, if so, what the changes are. As we shall see, this type of service can be provided for *website* browsing. On the other hand, it is obviously not possible to provide such information for the entire Web.

Using the agent technology, we design a friendly user interface that uses *personalization* techniques to help a user navigate through a website. In the following subsections, we first examine problems related to website browsing and propose solutions for each for each of them. We then discuss, in Section 2.2, how a website browsing agent may utilize user profiles to improve performance. In Section 2.3 we propose to use a dedicated window as the interface between user and agent.

2.1 Problems and Solutions

A basic assumption of website browsing is that the user is serious about obtaining useful information through browsing[1]. To help the users navigate through a website, we classify users into two types, those who are familiar with the website (both content and functions) and those who are not. Different types of users may have different problems and needs. For instance, a first time user may need information on the content and layout of the website, while an advanced user may need more sophisticated mechanisms to assist in navigation. In Table 1, we list problems that are usually encountered in website browsing.

From Table 1 we can see that different types of users need different kinds of assistance. The most important help a first-time user needs is to get a "feel" about the website, while a user who is already familiar with the website wants to get things done more efficiently.

Since it is difficult in practice to decide if a user is new to the website or not, we use a *registration* approach to distinguish the two classes of users. An unregistered user will be regarded as an unfamiliar one; and a registered user will be classified as either familiar or unfamiliar depending on her profile (issues concerning user profile will be discussed in Section 2.2). An unregistered user may register in order to acquire more assistance.

In the following, we discuss each problem and propose practical solutions which will be further elaborated through the case study in Section 3.

- **What kind of information the website contains?**
  When a user visits a website for the first time, the first question she may raise is whether the site contains anything of interest. A simple way to solve this problem is to create a page with an overview of the site and to add a hyperlink in the homepage which points to it. This hyperlink should be placed at an emphatic position so that a casual user can find it without too much effort.

[1]One cannot make the same assumption with web browsing.

User Type	Problems
Unfamiliar	- What kind of information the website contains? - How to find pages likely to be interesting? - How to avoid getting lost when navigating through the website?
Familiar	- Is there something new? - How to find useful pages more efficient? - Are there useful tools to help with reading?

Table 1: Website browsing problems for two kinds of users: Those who are unfamiliar or familiar with the website.

It may even be better to introduce the site content in a more active way. For instance, the website designer may collect a set of *demo pages* and asks an *demo subagent* to present the pages to the user upon request. This feature is similar to a slide show.

- **How to find pages likely to be interesting?**
  Once a user decides that the website is of interest to her, she needs a way to find a few pages which are indeed interesting. The demo subagent is helpful in this regard. If the user is interested in a demo page, she may ask the demo subagent to mark down its URL to explore further later. It is also always useful to have a *website search engine* which is tailored to searching the content of the website and can take advantage of the metadata, if any, that are associated with the content.

  We should emphasize here that we believe *every serious website should have a website search engine*. The ineffectiveness of generic search engines on the Web is widely acknowledged. Unless a global metadata system can be established and widely adopted, it is hard to see how generic search engines can significantly increase their precision rate, not to mention recall. A website search engine, on the other hand, can effectively utilize the unique properties embedded in the local data, and thus achieve much better precision and recall. It is a challenging problem to build general search machinery which can be quickly incorporated into a specific website.

  Another useful feature is a list of FAQ's (frequently asked questions). While FAQ's have been used in newsgroups for years, not many websites have taken advantage of this concept in getting people acquainted with their content.

- **How to avoid getting lost when navigating through the website?**
  It is easy to get lost after clicking many hyperlinks. One way to solve this problem is to provide a *site map* which tells the user where she is. Conceptually, the map works like a traditional two-dimensional roadmap which identifies the location of the user. However, since website content is usually connected via hyperlinks, it is easier to construct a text-based map which hierarchically organizes content in the site. The position of the page being browsed can be highlighted. A *trace map* is a map which describes what the user has visited. It can take the form of bookmarks or history links which are provided by conventional Web browsers. Combining a site map and a trace map allows the user to have some idea about what she has done and has missed.

  Analyzing revisitation patterns is also helpful for website navigation [7]. For instance, if certain website pages are often revisited by a user, URLs linked to these pages may be presented as shortcuts so that the user can connect to these pages without too much efforts.

- **Is there something new?**
  When a user revisits the website, it is usually beneficial to tell her if the website content has been changed or updated since the last visit. This can be done by keeping a personal profile in the website which records the user preferences. When logged in, an agent is invoked to examine the personal profile, check for new material in the website, and notify the user whether there is any new content that may be interesting.

- **How to find useful pages more efficiently?**

Many content-rich websites are designed based on some methodology, in the sense that there is some consistency on the layout, how material is presented, and how pages are constructed. Many also employ metadata in organizing information. Such characteristics can be utilized in the website search engine to provide much more effective (and precise) search. The metadata can be further used to classify website content into categories, and to classify and organize the returned query results. For instance, if date is part of the metadata, query results can be arranged chronologically, or even grouped according to the year.

We remark again that it is more difficult for generic search engines to have similar features because the webpages that a generic search engine needs to handle may be very disorganized. Search engines such as YAHOO! do a fairly good job in classifying webpages. But it is done with great expenses and tremendous manpower. There are traditional IR techniques which can classify webpages automatically, but the results are usually non-intuitive and difficult to use.

Sometimes a user may want to associate two pages in some way. For instance, if the user is reading a page p, and wants to find those that she has already visited which are related. This can be done by providing a "user history" subagent which helps the user find visited pages that are relevant to p. We shall describe a "user history" subagent in Section 2.2.2.

- **Are there useful tools to help with reading?**
  Reading-aides, such as dictionaries, encyclopedia, calendar converters, and language translators, are always helpful when reading webpages. Since these tools are expensive to build, it is impractical to expect all websites to provide such tools. Fortunately, many such tools are already available on the Web. Thus, all that needs to be done is for a website to find the most useful tools and build a *broker agent* which connects to the tools and process the user demands.

  To give some examples of services that one can find on the Web, the service provider FOLDOC [2] is an on-line dictionary of computing; the Internet Encyclopedia of Philosophy [3] is an on-line encyclopedia of philosophy; and the Calendar Conversion program [4] which converts among at least 14 different calendars [3]. While the encyclopedia help with defining terminology, the calendar converter can be extremely useful for a website about history. As shown in Figure 1, an agent may act as an *information broker* between a user and a service provider. In Section 3.3, we shall discuss in more detail such an information broker.

  Another useful tool is a chatroom facility which allow users to communicate on line and discuss about the details of the website.

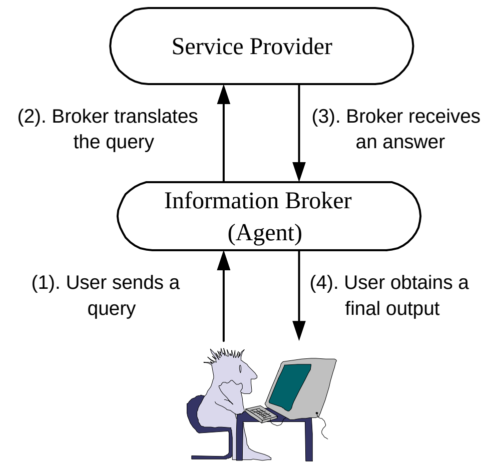

Service Provider
(2). Broker translates the query
(3). Broker receives an answer
Information Broker (Agent)
(1). User sends a query
(4). User obtains a final output

Figure 1: The process flow of an agent as an information broker.

We summarize the proposed solutions in Table 2. Since a user demands various kinds of services when browsing a website, it is desirable to integrate these services in a uniform way. In Section 2.3, we propose to use a dedicated window approach to integrate all agent functions.

2.2 Personalization and User Profiles

One of the most challenging tasks for agent design is the notion of *personalization*. Users often have dif-

[2]http://wombat.doc.ic.ac.uk/foldoc/index.html

[3]http://www.utm.edu/research/iep/

[4]http://emr.cs.uiuc.edu/reingold/calcode.shtml

User Type	Proposed Solutions
Unfamiliar	- An overview page of the website content. - An agent which provides a slide show. - An agent which suggested most frequently visited pages. - An FAQ about the website. - A site map and a trace graph for navigation.
Familiar	- A registration facility to keep a user's personal profile. - An agent which keeps user updated on the changes since last visit. - A website search engine for precise retrieval. - An agent which categories visited pages and classifies query results. - An information broker which serves as a reading-aide. - A chatroom which allows a user to communicate with others.

Table 2: Proposed solutions to the website browsing problems of Table 1.

ferent preferences (e.g., page style, emphasis, reading tendency) about website pages. A good website should have ways to adapt to different users.

The difficulty of personalization comes from the fact that a user's preference may vary with time and with topic. For simplicity, however, it had often been assumed that the preference is fixed and can be learned from a set of training examples. In this paper, we call *user profile* the data structure which maintains attributes concerning user preferences. If a profile is about a single person, we call it a *personal profile*; otherwise, we call it a *group profile*. From an agent point of view, a personal (group) profile corresponds to the knowledge about the preference of a user (or a group of users respectively).

User profiles are important for an agent to achieve personalization. In the following subsections, we investigate how a website search engine and a "what's new" agent may employ user profiles to improve performance.

2.2.1 Profiles for the Website Search Engine

A website search engine finds pages within the website (called *website pages*) for a given user query. In order to help a user formulate appropriate queries, it is usually helpful for a website search engine to suggest candidate queries. This idea can be done by making use of profiles which contain information about past queries (of the current and other users) and about website pages recorded as interesting from past record. Let U be the set of all users [5] and Q be the set of all possible queries[6]. Given $q \in Q$ and $u \in U$, let $x_{u,q}$ be the number of query q issued by the user u, and let R_u be the set of Web pages marked as interesting to u. Let $n_q = \sum_{u \in U} x_{u,q}$ count the total number of queries q issued by all users. We may regard the tuple $\langle \{x_{u,q}\}_{q \in Q}, R_u \rangle$ as the personal profile of user u, and the collection $\{n_q\}_{q \in Q}$ as the group profile of all users.

Given the information $\{n_q\}_{q \in Q}$, the search engine may rank queries in Q in decreasing order of the number of requests issued. Let Q_n be the first n ranked queries. Intuitively, Q_n contains the top-n popular queries issued by all users. For an unfamiliar user, the website search engine may use Q_m (say $m = 10$) queries as suggestions. For a familiar user (i.e., a registered user u whose total queries $\sum_{q \in Q} x_{u,q}$ exceeds some threshold), the search engine may use information from R_u to further improve suggesting queries. For instance, we may employ the technique of *relevance feedback* [9], which is widely accepted by IR researchers, to extract significant terms from a set of text-based documents. Let S_n be the top-n ranked significant terms of R_u. The website search engine may then suggest $Q_m \cup S_m$ (say $m = 5$) to a familiar user.

[5]We regard all unregistered users as a single, special one.
[6]Q is, obvious, a conceptual set.

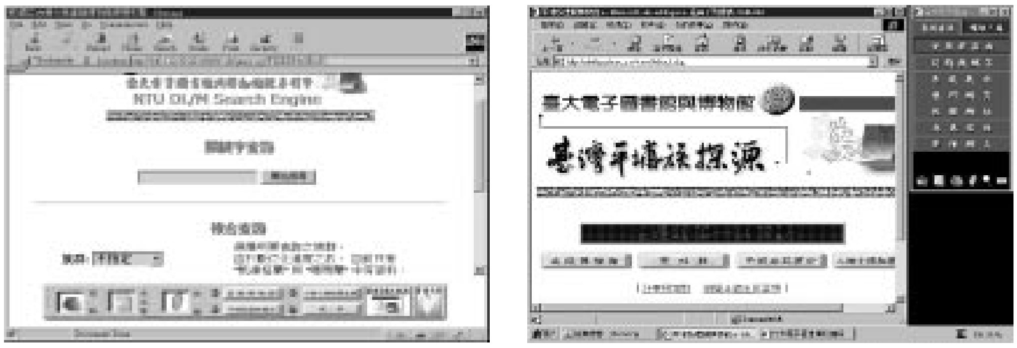


Figure 2: A dedicated control frame (the bottom portion in the left picture) vs. a dedicated control window (the upper-right window in the right picture).

2.2.2 Profiles for the "What's New" Subagents

The content in a website may change. Although the website may contain a large database from which most of its information is drawn, we may still regard the website content as website pages, since the website must present the content in a browser-readable form so that a user can read the content from an Internet browser. Let D_n be the website pages which are *added* within the last n days, then a "what's new" subagent is responsible to return D_n, for some predefined n, to a user.

Assuming that $n = 30$ and that a registered user had visited the subagent one week ago. Then obviously it is more reasonable to simply report what had been added within the last 7 days. To achieve this goal, the subagent needs to simply maintain a number t_u in the user's personal profile which indicates that the user u did not check for new content in the last t_u days. Then the subagent may return D_x, where $x = \min(30, t_u)$, as new website content.

This straightforward solution is not satisfactory if a massive amount of information had been added recently. We propose two possible approaches to this problem. If a new collection has been added, the subagent can simply provide the user with a link to the homepage of the new collection. Concurrently, the subagent can also utilize the user's preference. Assume that the website pages have been classified into several categories, and that the user's preference identifies one or several of the categories as being interesting, then only the new website pages that fall into those categories need be reported.

What was described above can be done as follows. Given a user u, let C_u be the set of categories preferred by u and let V_u be the set of website pages visited by u. We denote by $V_u(c) \subseteq V_u$ the set of visited pages that are in the category c. The set V_u can be obtained by keeping a record of which website pages have been visited. The set of interesting categories C_u can be obtained either by asking a user to subscribe to interesting categories, or by employing learning techniques to the set V_u [?]. For instance, we may regard a category c as interesting to user u if $|V_u(c)|$ exceeds a certain threshold. Once the personal profile $\langle t_u, C_u, V_u \rangle$ is obtained, new website pages in category c, where $c \in C_u$, can be highlighted as new content likely to be interesting.

The information of categories and visited pages can also be applied to the "user history" subagent. As discussed in Section 2.1, a user may want to find previously visited pages which are relevant to the current page. Let p be the current website page and c be its corresponding category. The subagent may suggest $V_u(c)$ as visited pages relevant to p.

2.3 Integrated User Interface

All functions offered by a website agent should be organized in a uniform interface they can be accessed easily. This is usually done by using *frames*, where the browsing window is separated into two frames, one (the control frame) is dedicated to important hyperlinks or agent functions, and the other (the content frame) is used to display normal page content. (See Figure 2 for an example.) While it is easy to invoke website functions using frames, they also suffer some serious problems. For instance, one cannot bookmark the current page and returns to it later; URLs stop working; and printouts become difficult [6]. An even more serious problem is that if one accidentally follows an external link and connects to another website, then the control frame will also disappear. This way the user may lose the agent simply because she has casually visited a page not belonging to the website.

In our approach we use a *dedicated control window* to manage the website functions. A dedicated control window is a more appropriate agent interface. Choosing a unction from a dedicated window is as easy as using a frame, but it is more intuitive to open an auxiliary window (say, to perform chatroom functions or search facilities) from the control window than from the control frame. A user may open the control window to acquire agent assistance and may close the window when the service is no loner wanted; and one may minimize the control window to have a larger screen area to display content.

3 A Website Browsing Agent for the NTU Digital Library

In this section we describe an implementation of a website browsing agent for the National Taiwan University Digital Library and Museum (NTUDLM).

NTUDLM is a project supported by NTU and the National Science Council of Taiwan for building a digitized repository of historical archives and artifacts concerning the indigenous Taiwanese people and earlier settlements. It is accessible through the Web at HTTP://NTUDLM.NTU.EDU.TW. The goals of the project are to build a system which

1. provides a user-friendly Web-based environment for learning about Taiwanese history,
2. provides a modern research tool for historians and anthropologists of Taiwan,
3. preserves Taiwanese history and artifacts.

Our browsing agent was built with the languages JAVASCRIPT, JAVA and CGI programs (written in C), and works with NETSCAPE 4.02, NETSCAPE 4.05, IE 3.0 and IE 4.0.

3.1 Functions Offered by the Agent

We have implemented most of the functions described in the previous sections. For the ease of access, we divide these functions into three main groups:

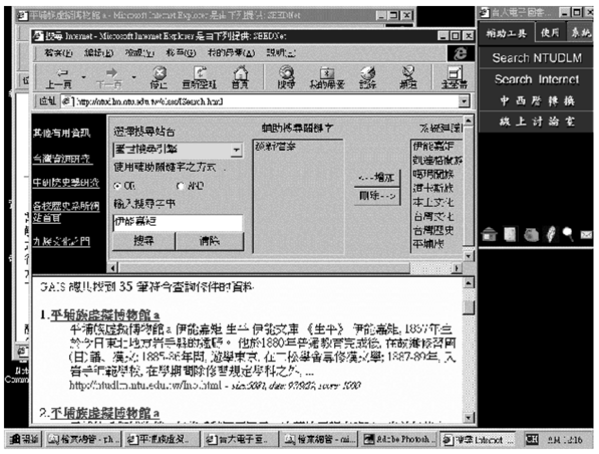


Figure 3: An illustration of the "Web search" subagent. The subagent works as a meta-search engine which answers user input with the help of existing Web search engines.

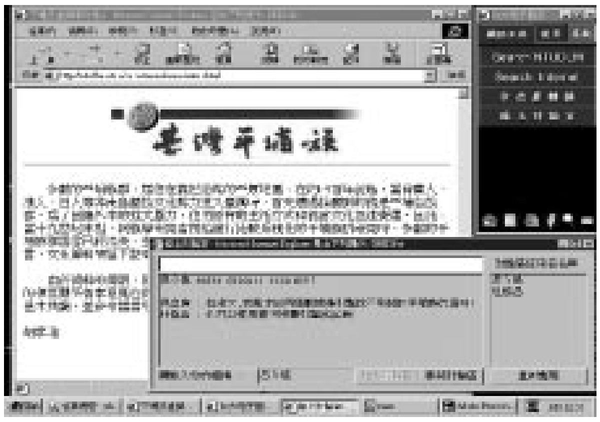


Figure 4: An illustration of the chatroom function.

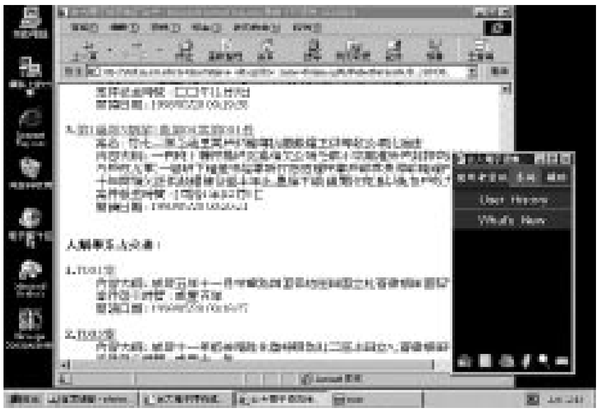


Figure 5: An example of the "user history" subagent.

- System resources: This group contains six services about system resources. They are:
  1. An introductory page about the content of NTUDLM and about how to use the agent.
  2. An FAQ page which answers questions frequently asked by users.
  3. A demo subagent which presents selected NTUDLM pages automatically. A user may change the time required to show a webpage and may stop the demo at any time.
  4. A page with a list of hot website pages (i.e., most frequented pages).
  5. A page which collects Web pages or resources related to the NTUDLM.
  6. A mailbox which allows a user to ask questions by e-mail. The messages will be answered by the maintainers of NTUDLM.
- Auxiliary tools: This collection contains services to help a user find or read Web pages. It consists of four subagents:
  1. A local search engine which helps a user find website pages. We employ techniques discussed in Section 2.2.1 to improve the search performance.
  2. A meta-search engine which utilizes general-purposed Web search engines to find (external) Web pages likely to be relevant to a user query. We illustrate an example in Figure 3.

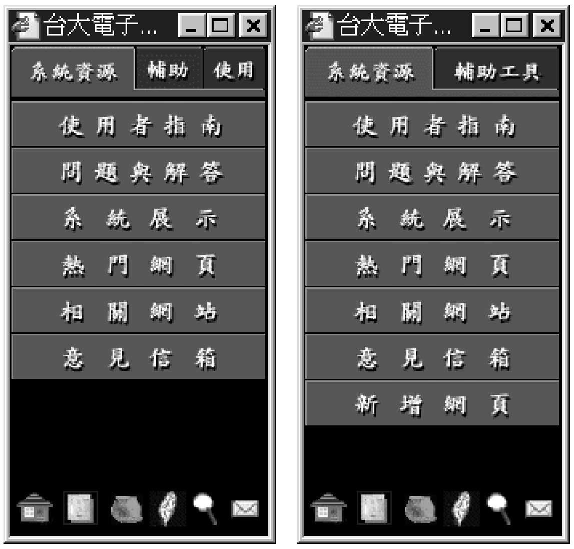


Figure 6: Control windows for a registered and a unregistered user. There are three groups of agent services for a registered user (left window)but only two for an unregistered one (right window).

  3. A subagent which uses a calendar translator[7] to help a user convert between Chinese and western calendars. In Section 3.3, we shall discuss the subagent in more detail.
  4. A chatroom (Figure 4) which allows users to communicate on line.
- User profile: Currently, there are only two subagents which offer personalization services for a registered user. (We have not yet implemented the site map feature, partly because NTUDLM is not yet fully completed.) Both subagents utilize the profile techniques discussed in Section 2.2.2:
  1. User history: A subagent which maintains what a user has been visited. Visited pages that are in the same category as the current browsing page will be ranked higher (cf. the last paragraph of Section 2.2.2). See Figure 5 for an illustration.
  2. What's new: A subagent which notifies a user whether there are new NTUDLM content likely to be interesting. For an unregistered user, the "what's new" service

[7]`http://www.sinica.edu.tw/ftms/luso.htm`

is moved to the "system resource" group since the subagent does not have a profile about the user (cf. Figure 6).

3.2 Registration Process

As discussed in Section 2.1, a website needs to know whether a user is registered in order to provide agent assistance. In our implementation, we use the following mechanism to obtain such information. When a user loads the NTUDLM homepage, a JavaScript program (embedded in the HTML source of the homepage) and a Java program will execute automatically. The JavaScript program checks whether there is a *cookie* (stored in the local disk) indicating previous registration record, and opens a control window to offer agent services. As shown in Figure 6, if the user is registered, the control window will display three tabs (each contains a group of functions discussed in the previous section). If the user is unregistered, then there will be only two. What is missing contains those functions that need user information. The Java program, in the meantime, displays a welcome panel. If the user is unregistered, the panel displays a message inviting the user to register. A user may skip registration if she wants. The process of checking registration to obtain the assistance window is illustrated in Figure 7.

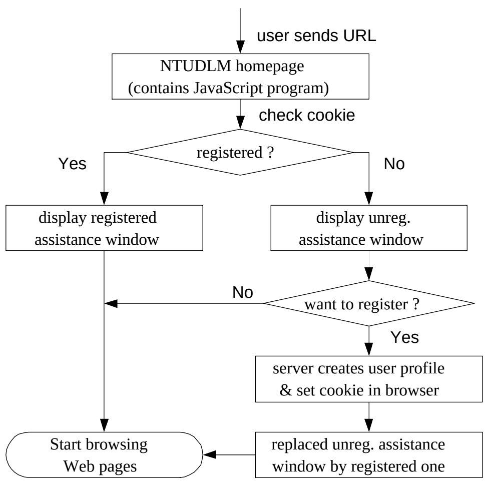

user sends URL
NTUDLM homepage (contains JavaScript program)
check cookie
registered ?
Yes
No
display registered assistance window
display unreg. assistance window
No
want to register ?
Yes
server creates user profile & set cookie in browser
Start browsing Web pages
replaced unreg. assistance window by registered one

Figure 7: The process for a user to obtain assistance window.

3.3 Subagent as an Information Broker

In this section, we demonstrate the information broker feature by describing the implementation of a subagent which offers the service to translate between the Chinese calendar and the Gregorian. When a user inputs a Chinese (or Western) date[8], the subagent will return the translation result.

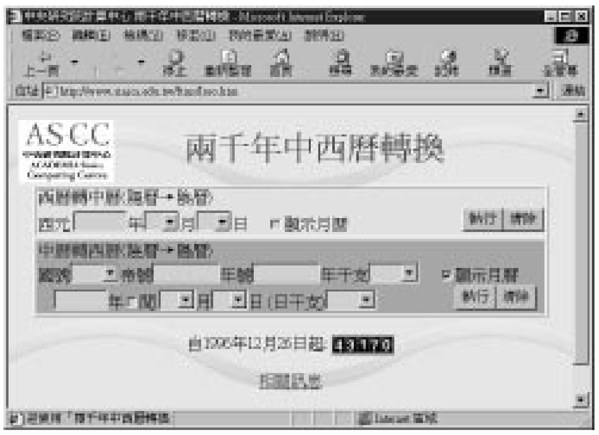
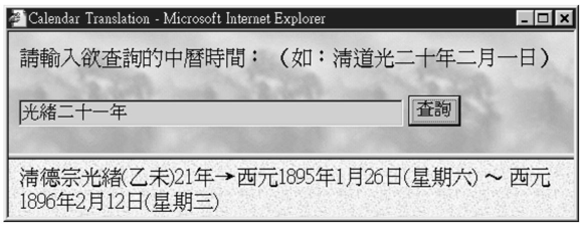


Figure 8: The calendar translator Web page. A user fills entries in the table and receive a calendar-translation result.



Calendar Translation - Microsoft Internet Explorer

請輸入欲查詢的中曆時間：（如：清道光二十年二月一日）

光緒二十一年 查詢

清德宗光緒(乙未)21年→西元1895年1月26日(星期六) ~ 西元1896年2月12日(星期三)

Figure 9: The "calendar translation" converts a Chinese date into the corresponding Gregorian date.

Since there is already a calendar translator (Figure 8) available on the Web, it is unnecessary for the subagent to do the conversion itself. For instance, our "calendar translation" subagent simply accepts a user input, parses and transforms to a form accept-

[8]The original idea is to use drag-and-drop to reduce user effort. A user may mark a string as the input and then drag-drop to the subagent button to obtain the translation result. However, it seems difficult to implement such mechanism in the current JavaScript language.

able by the calendar translator, receives output from the calendar translator, and then reports the result to the user. With the parsing process, the subagent checks the user input to see whether it is ambiguous or illegal. If such an error occurs, the subagent prompts a message to ask the user to modify the input. We illustrate a subagent output in Figure 9.

It is conceivable that there will be more and more Web pages which offer question-answering services. The approach to regard subagents as information brokers is practical and desirable, since it saves a lot of programming effort to offer agent assistance.

4 Concluding Remarks

The most useful information on the Web usually comes from carefully designed websites devoted to a specific topic. Such websites deserve special tools to help with browsing and navigation.

Website browsing is different from *Web* browsing in that a content-rich website is usually much better structured and its information content is better organized. There is usually structural information, such as metadata, that can be used to facilitate search and retrieval. We pointed out *personalization* as the main issue that should be solved in website browsing, and advocated that every content-rich website should have its own *website search engine*, which retrieves data within the website. We also listed problems with website browsing and navigation that are pertinent to users who are familiar or unfamiliar with the website, and proposed to use agent technology to solve them. We demonstrate how our idea works by presenting a working website browsing agent which assists people in navigating the NTU Digital Library and Museum. This agent was implemented in JAVA, JAVASCRIPT and C, and it works with both NETSCAPE COMMUNICATOR and MICROSOFT INTERNET EXPLORER.

What we proposed can also be applied to the Intranet environment, in particular for managing data of big corporations which are built for internal use probably in the form of a digital library.

There are many research problems that still need to be solved. For instance, better methods are still needed to capture user behavior on website navigation. Furthermore, since metadata plays an important role in categorization and retrieval, there should be automatic or semi-automatic ways to help website content builder to add or modify metadata about website content. There should also be a methodology for building search engines which can be quickly adapted to build website search engines.

References

[1] Rob Barrett, Paul P. Maglio and Daniel C. Kellem, *How to Personalize the Web*, ACM SIGIR'97, 1997. Available via `http://www.acm.org/sigchi/chi97/proceedings/paper/rcb-wbi.htm`.

[2] Fah-Chun Cheong, *Internet Agents: Spiders, Wanderers, Brokers, and Bots*, New Riders Publishing, Indianapolis, Indiana, 1996.

[3] Nachum Dershowitz and Edward Reingold, *Calendrical Calculation*, Cambridge University Press, Cambridge, 1997.

[4] Henry Lieberman, *Autonomous Interface Agents*, ACM SIGCHI'97, 1997. Available via `http://www.acm.org/sigchi/chi97/proceedings/paper/hl.htm`.

[5] Michael L. Lyu, *The Design and Implementation of Integrated User Interface for Web Browsing*, Master Thesis, National Taiwan University, 1998.

[6] Jakob Nielsen, *User Interface Design for the WWW*, ACM SIGCHI'97, 1997. Available via `http://www.acm.org/sigchi/chi97/proceedings/tutorial/jh.htm`.

[7] Linda Tauscher and Saul Greenberg, *Revisitation Patterns in World Wide Web Navigation*, ACM SIGCHI'97, 1997. Available via `http://www.acm.org/sigchi/chi97/proceedings/paper/sg.htm`.

[8] Hsieh-Chang Tu and Jieh Hsiang, *An Architecture and Category Knowledge for Intelligent Information Retrieval*, The 31^{th} Hawaii International Conference, HICSS 31, 1998.

[9] Bienvenido Vélez, Ron Weiss, Mark A. Sheldon and David K. Gifford, *Fast and Effective Query Refinement*, ACM SIGIR'97, pp. 6-15, 1997.

[10] Michael Wooldridge and Nicholas R. Jennings, *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review, Vol. 10, No. 2, June 1995.