PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :
G06F 13/00, 13/14, 17/00, 17/30, 17/60

A1

(11) International Publication Number: **WO 00/55741**

(43) International Publication Date: 21 September 2000 (21.09.00)

(21) International Application Number: PCT/US00/04588

(22) International Filing Date: 23 February 2000 (23.02.00)

(30) Priority Data:
60/125,048 18 March 1999 (18.03.99) US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application
US 60/125,048 (CON)
Filed on 18 March 1999 (18.03.99)

(71) Applicant *(for all designated States except US)*: BLINK.COM, INC. [US/US]; 64 Fulton Street, 6th floor, New York, NY 10038 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: SIEGEL, David [US/US]; 401 East 80th Street, Apartment 8G, New York, NY 10021 (US).

(74) Agents: SWEENEY, John, F. et al.; Morgan & Finnegan, L.L.P., 345 Park Avenue, New York, NY 10154 (US).

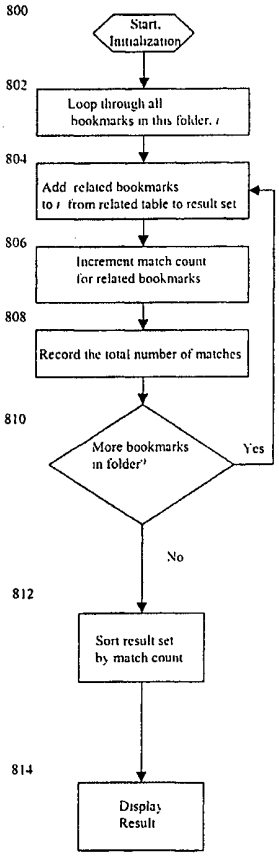
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: SYNERGISTIC INTERNET BOOKMARKS COMBINING INTERNET SEARCHING AND HOT LINKING

(57) Abstract

In various aspects, it is among the objects of the present invention to provide a system and method for managing and automatically categorizing World Wide Web links (also known as URLs, or universal resource locators) to create personal and public directories of these links to increase the navigability of the Internet. In various embodiments, the system of the present invention stores web links for multiple users in a database and provides methods for extracting and displaying the web links, methods for finding web links that are related to existing links that a user has stored in the system, and other related features (804). Users can organize and manage collections of links and to find related links in other user link collections (808). The database matching feature can use other information for finding relationships, such as profile information on the user, including, e.g., age, gender, and occupational profession.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYNERGISTIC INTERNET BOOKMARKS COMBINING INTERNET SEARCHING AND HOT LINKING

Cross-Reference to Related Applications

This application is related to and claims the benefit of priority to U.S. provisional patent application serial number 60/125,048.

Technical Field of the Invention

The present invention relates generally to access of internet information and more particularly to facilitating automatic management of access to web pages.

Authorization With Respect To Copyrights

Background

The Internet World Wide Web, or simply "web," is a place where people store and distribute information. The usefulness of the web depends a great deal on people's ability to locate material relevant to their interests. As the web has matured, the amount of information available on the web has grown dramatically resulting in increasing difficulty of finding information only relevant to a particular topic of interest.

The web is essentially a collection of linked content pages that can be imagined to resemble a spider's web. The usefulness of the web is, to a large extent, determined by how easily information stored in one place can be located by someone in another. Information stored on a Web page is typically accessed through a program called a web browser (e.g., Netscape or Internet Explorer) via a "Universal Resource Locator" or "URL". The URL is commonly referred to as a "web address", a "hyperlink", or simply a "link". An example of a URL is http://www.ibm.com/. When a URL is entered into the browser on a local computer, the browser will connect to a web server and display a particular page of content associated with that URL. Web pages can further include embedded

hyperlinks which, if selected in some way (e.g., by clicking with a mouse), will transfer the browser to the corresponding page in similar fashion.

As the Internet has evolved, so have the methods for finding relevant content so that the hyperlinks are useful. Current primary methods for managing links or web directories in use today include:

- link lists that a user can store locally in his computer and access with his browser;
- locally stored history files that contain a list, organized by date and time, of sites that a user has visited;
- link lists that are manually collected and placed on web pages (e.g., the home page of an organization);
- web link rings, that link together pages of related content; and
- search engines that catalog all web links and allow a user to perform a search for some specified content.

Many web pages contain list of links, often collected into various subjects. Since navigation from one location to another on the web is generally accomplished by either typing a URL into a browser, or by selecting a URL from a list of one or more links, the location of material is typically presented to a user as a list of hyperlinks. These links are commonly called "hot lists".

Also common are automatic web "search engines" like AltaVista and Infoseek, which automatically and continuously scan the Web for all content and build a database of addresses and keywords. A user can search the AltaVista database for potentially relevant content. These automatic web search engines perform textual matches against a set of keywords that a user provides in their query. Pages that simply contain the text, not necessarily pages that are relevant to the concept sought for, are found. A resulting problem is that, in an effort to generate site hits, many Web page authors attempt to fool search engines into returning their pages as a match for a search when the user performs a common search for something else. Most users of search engines have experienced this problem. For example, a web site for a book store might insert words in their home page like "white house", in hope of getting their link displayed when someone performs a search trying to find the home page of the White House.

Sites such as Yahoo and the Mining Company maintain large collections of organized links which are manually compiled and grouped into categories and made available to users.

Finally, most browsers have a mechanism that allows the user to manually save the addresses of their favorite web pages. This feature is commonly called "bookmarks" or "favorites". Typically when users navigate the web and they find a page with contents that interests, them they "bookmark" it or add it to their "favorites" so that they can easily return to it without having to search for it again. This often results in disorganized and hard to use personal bookmark collections when all the bookmarks in left one unorganized grouping. At some later point, they may sort the bookmarks and put them into related folders. For example, when using the bookmark feature in Internet Explorer or Netscape, a user has to search for the right folder to file a new link in the (potentially large) collection of folders that the user has already created. If a relevant folder cannot be found, the user must determine an existing folder that a new subfolder should be created in to store the new link. As the user develops a larger bookmark collection, it will take increasingly more time to organize new entries and determine the best place to store the link.

Summary and Objects of the Invention

In various aspects, it is among the objects of the present invention to provide a system and method for managing and automatically categorizing World Wide Web links (also known as URLs, or universal resource locators) to create personal and public directories of these links to increase the navigability of the Internet. In various embodiments, the system of the present invention stores web links for multiple users in a database and provides methods for extracting and displaying the web links, methods for finding web links that are related to existing links that a user has stored in the system, and other related features.

The system and method of the present invention allows users to organize and manage collections of links and to find related links in other user link collections. The database matching feature can use other information for finding relationships, such as profile information on the user, including, e.g., age, gender, and occupational profession. In essence, all participating users of the system help to

organize a web directory.

The present invention produces lists of links that are self-organizing into clusters of related sites based upon the collective organizational efforts of all users of the system. Unlike conventional systems, it is neither keyword nor textual searching based. The information implicit in an individual's clustering of related web links is used to create a searching and clustering mechanism that allows users to find content that is related to existing content that they have already found. The present invention offers ease of use and does not require users to make explicit categorization decisions. This approach creates a self-organizing web directory that takes advantage of the experiences of all participating users.

The self organizing link list of the present invention: allows users of the World Wide Web to organize their own web links to content that they find interesting and to locate interesting and relevant content with current search engines; allows more efficient link management; are more specific, and less prone to false matches than the results of conventional keyword-based search engines; are formed without manual intervention; reflect the actual preferences and usage patterns of users, and are relatively free from the editorial bias or knowledge limitations of the human editors of catalog search sites; and, reflect the collective efforts of other users who have found interesting relevant content and provides a systematic approach to share their findings. Various embodiments of the invention may offer one or more of these advantages.

Brief Description of the Drawings

The foregoing and other features and advantages of the present invention will become more apparent in light of the following detailed description of exemplary embodiments thereof, as illustrated in the accompanying drawings, where

Figure 1 illustrates a system architecture implementation of the preferred embodiment of the present invention;

Figure 2 illustrates a flowchart of the main processing loop according to the preferred embodiment of the present invention;

Figure 3 illustrates an exemplary main screen according to the preferred embodiment of the present invention;

Figures 4A and 4B illustrate user definition and folder mapping database schema under the preferred embodiment of the present invention;

Figure 5 illustrates a folder matrix derived for an illustrative number of links under the preferred embodiment of the present invention;

Figure 6 illustrates a link scoring table demonstrating link scoring under the preferred embodiment of the present invention;

Figure 7 illustrates a link relation result table derived under the preferred embodiment of the present invention;

Figure 8 illustrates a flowchart of a preferred implementation of the matching function of the present invention; and,

Figure 9 illustrates a flowchart of a preferred implementation of related link processing of the present invention.

Detailed Description

In a preferred embodiment of the architecture shown in Figure 1, each individual Web User 1, 2 through *n* (e.g., each user via a PC with internet access capabilities) initiates a session with a central server computer 10, preferably via a local web browser (e.g., Netscape or Internet Explorer). At a central site 100, the central server computer 10 accesses data storage device 20 which includes a database. In the preferred embodiment, users will access the present invention by a PC via the public Internet. However, in alternative embodiments, implementations of the present invention can take place over a private Intranet or the implementation can fully avail itself of alternative technologies for web users access, e.g., cellular telephone access, Web TV™ etc. To achieve optimal performance, the system can, in alternative embodiments, be operated on more than one central computer and more than one database. Conventional software solutions are utilized to keep the multiple copies of the database synchronized, and to distribute the load of web traffic among more than one server.

In the present invention, the "favorites" or "bookmarks" function of the web browser is centralized at the single common site 100. All users of the service store their bookmarks at the common central site 100, rather than using the decentralized (local) bookmark feature of their own browsers. As will be discussed in further detail *infra*, bookmark collections or lists created by individual users and

stored in database 20 at the central site are periodically analyzed through comparison of list pairings. When two lists are found that are highly correlated, those two lists are deemed related and a new, centralized list is generated as a ranked, complete or partial union of the two original lists. Other users of the system, as well as the owners of the two original lists, will now access and use this new centralized list. It gives a more comprehensive list of related sites than either of the original lists alone.

This process allows users of the Internet to search for and group URL bookmarks (links) into a hierarchical set of folders that are stored on a central server and that can be accessed from any web browser on the Internet. Based on the collective results of individual user folder groupings, relationships between links are automatically found, helping users to locate relevant information on the Internet. As an example, the process will propose links to the user that are similar to the links that the user has previously collected in a folder. Other links will be identified and rank ordered links on the basis of a measure of similarity based on a number of criteria including the number of other users that grouped them in a similar way, a distance metric that relates links by intersecting the link content of user folders, user provided ratings, and by profile information on the user's background.

Web users access the present invention by connecting to a designated web site. An illustrative example of the main system flow path according to the preferred embodiment is shown in Figure 2. User login 200 initiates the flow where a username and password is either entered upon query into the system directly from the web based application, or passed automatically via "cookies" or other browser provided features. The system validates 202-204 this information against the user account database. Where the user is not a registered user, a registration procedure is undertaken 203, the user is registered and a user account is created 213 and the flow continues to 206. In the event of an invalid username/password combination, an error message is posted 205 and the user is asked to start again at user login 200. If the username and password match valid entries in the database, the system starts 206 a new web session for the user at the user's home page, and generates (or displays) 208 a custom home page directory that lists the top-level links and directories set associated with the user. The displayed page (and all subsequent

pages) in this embodiment (see Figure 3 *infra*) contains links 30 (see Figure 3 *infra*) or buttons 32 (see Figure 3 *infra*) that allow the user to perform various link-related functions. For example, the user can click on a link (i.e., a bookmark) 210 to go to the corresponding web page 211 and update a hit count 212 corresponding to that site; or the user can click on a folder 214 to display the corresponding folder content 215, which may include more links and folders; or, the user can alternatively click on a given set of commands 216 to, e.g., perform 217 the corresponding operation such as: navigating to a subdirectory (if one exists), and moving back to a previous directory in the directory hierarchy; adding a new link to the current directory; and, deleting a link from the current directory.

This organization of links is similar to a standard computer file system, organized into a hierarchical structure of files and folders. While similar to the approach used in many web-based directories, such as Yahoo, in a site like Yahoo, there is one web directory which is the same for everyone. It is the only directory that a Yahoo user can navigate. To the contrary, when a user clicks on a link in the system of the present invention, the system records in an entry for the selected link in the user's database the date and time, and a counter is incremented to keep track of the total number of times the user has clicked on the link. That same information is then added to an aggregated database with the pooled information of all users for that link. Thus, the system keeps track of the total number of visits to a link for each individual and also for all users. Next the system will transfer web control to the selected page and the web browser will display the selected page (possibly in a new window).

An example of a screen layout according to this embodiment of the present invention is illustrated in Figure 3. The main screen will provide various links 30 and buttons 32 implementing the various features discussed with respect to Figure 2 which links and buttons can be clicked to, e.g., go to a link or add new links and folders. The "add new link" screen accessed by clicking the Add button 32 has fields in which to enter the name of the link, a description, and a rating. The "add new folder" field allows the user to specify the name of a folder, a description, and to indicate if the folder is "public". Public folders can be referenced and viewed by other users, and provide a method for sharing links.

The system will track the number of users that have referenced each public folder to determine, among other things, the popularity of the folder. For ease of use, an optional browser "add on" allows automatic insertion of the currently displayed page as well. When a new link is added, the system checks if any other users have previously added the link. If not, the system creates a new entry in the "database table" for the link and assigns to the link a unique "link id". The system then inserts an entry for this link into the current user link directory that is being displayed. Internally, the link will be referred to by the "link id", rather than with the text URL. This allows the system to easily identify all users that have referenced any link, along with other features to be later described. Once the link has been added, the hit count and the last access dates and times are initialized to 0 and "never" respectively. Subsequent clicks on the link, as previously described, generate updates of these fields.

Figure 4A illustrates an example of a user account database table for tracking users. Users of the system are given unique accounts that contain descriptive information about them. The information is stored in a database. While several categories of user information are shown in Figure 4A, the User Definition Table, other information can be dynamically added to this table to help classify the user as desired. A database table as illustrated in Figure 4B Folder Mapping Table maps folders to users. This user and folder information are used in the search and matching processes described *infra*.

Because access information and links are stored in a database on a per-user basis, various capabilities can be implemented to make it easier for the user to navigate through his or her link collection. One such capability is a search facility that finds links in the folders that the user has created. The search capability makes use of the histories of data, including the last date the link was referenced, the number of times the link has been accessed by the user and by all users of the system, the user-provided rating, and the descriptive textual information. The search capability also uses information obtained from other users who have added the same links to their collections. For example, a search can be performed to identify links from the user's collection that have the highest overall hit count across all users of the system. This would identify the most heavily used links in the user's

collection, system wide, which provides an indication that the links are in general interesting.

One of the important features of the present invention is an automatic categorization technique that groups together similar links. Links are collected in folders with arbitrary names that are created by each user. (A folder can, in turn, contain subfolders.) This is accomplished by taking advantage of the information that is collected implicitly when a user groups a set of links together into a single folder. The automatic categorization technique facilitates locating information on the Internet that might otherwise be difficult or impossible to find, in essence, by taking advantage of the collective body of users categorizing decisions. The categorization process relies only on the placement of links into a particular folder. Users do not assign categorizations to the links themselves.

Using categorization information implicit in the folder groupings overcomes a number of major shortcomings of conventional approaches based upon users manually assigning categories to a link. One such problem is that users will select different names for the same category. One person might assign a link for an online bookstore to the "shopping" category, another to "bookstores", and yet another to "bookstore". Non-English speaking users would likely select category names in their primary language. All of this would make it hard to relate categories together. Furthermore, getting a user to assign a category to a link requires a time consuming process that many users may choose not to perform. The implicit categorization approach based on the folder the user has placed the link into overcomes this problem.

A folder matrix (as illustrated in Figure 5) represents the collection of links stored in a particular folder. In Figure 5, the folders are represented by the columns of the table, and are labeled with letters A-F. The links in the system are represented by the rows, and are labeled with numbers 1-7. An "X" in a cell of the matrix indicates that the folder in the checked column contains the link in the checked row. In the example folder matrix of Figure 5, folder "A" contains links "1" and "2", as noted by the check marks that appear in cells A1 and A2. The link number in this table refers to a unique link ID present in the link database, in which additional information on that link is stored. The folders in the Figure 5 "Folder

Matrix" refer to folders created and maintained by one or more users of the system. In other words, folders "A" through "F" do not necessarily belong to one particular user. The ownership information for these folders is stored in a database table. For convenience, A1 is referred to as link 1 in folder A.

In this manner a database of links and folders are collected and managed on a per-user basis. That is, each user of the system (identified by a unique user account id) has a collection of such folders and links. The overall approach to using these collections globally to categorize the web links stored in the system is based on a rating of the similarity of folders. Similar folders are defined to be those that have similar link collections. For example, if many users are collecting links on vegetable gardens, it is to be expected that there will be overlap among the vegetable garden links that such users have collected. Not all collections of vegetable garden links will be the same. Some users will have links that most, if not all, of the other users will not have in their own folders. The system searches the link folders and, when presented with an exemplar folder, will find other links that are likely to be related to the links in the exemplar folder.

The folders in which different users of the system have stored links provides information as to the relationships between the links. For example, link 1 is related to link 2 because link 1 appears together with link 2 in both folders A and B. The users of the system who created folder A and B found it desirable to organize links 1 and 2 together, in the same folder. Since link 1 and 2 both appear together in folder A and B, the system considers them to have a direct relationship between each other. Links can also be related to each other via folders indirectly. Consider link 1 and link 4. In this case, there is no particular folder that contains both link 1 and link 4 (as there is no column in the table in Figure 5 that has an "X" for both those links). However, link 1 and 4 have an indirect relationship via link 3 in folder D, which contains link 4. Folder D also contains link 3. Link 3 is directly related to link 1 via folder B. Thus, link 1 and 4, via link 3, have a relationship between them. The creator of folder B had reason to organize links 1 and 3 together, to capture some relationship between them. The creator of folder D had some reason to related links 4 and 3. Thus, there is evidence of an indirect relationship between links 1, 3, and 4. Since different users of the systems create

folders and place links in them, the relationship between link 1 and 4 has been captured without any explicit categorization effort on the part of the user beyond the step of organizing their links into folders. The process described has no particular information on the nature of the relationship beyond the fact that users have placed them in related folders.

Returning to the above example, a user can have a folder with links on to sites with content related to vegetable gardens. There are two links in the folder, one for a web page on carrots, and another on tomatoes. A second user has a collection of links in a folder on carrots and peas. A third user has a collection of links on peas and peppers. In this collection of folders, no individual folders can relate the pepper link to the carrot link, since there is no a single folder containing both of these links. On the other hand, a path can be traced between carrots and peppers by the connection between carrots and peas in one of the folders. This indirect connection is not as strong as a direct relationship in which the links are all in one folder. Thus, the scoring of the relationship between the links in different folders will be smaller than the score of links in the same folder.

There are many uses for this categorization process. One important use is to enable users to find additional links that are related to links that they have already collected and stored in a folder. The user from the previous example may have created a folder that has a set of links that are about "gardening" or the folder may have more specific links, such as "vegetable gardens". The user clicks on a button that initiates a search and a list of relevant, related, links are displayed. The information is gathered by searching all link folders in the system, finding ones that appear to be similar to the user's "gardening" folder (the "exemplar folder"), and the best matches are presented. The same searching approach is used to find the best matching "public folders" to a user's exemplar folder. Again, the user clicks on a button and the system searches all public folders to find the ones that appear to be closest (but that contain a superset of the links that are in the user's folder). The details of the matching process are described below. The resulting matches are ordered by the popularity of the public folder, based on how many other users have connected it into their folder tree. If the user finds a particular public folder link collection useful, they click on the appropriate button that adds it, as a read-only

folder, to their link collection. Any changes that the folder owner makes to this folder are immediately reflected in all read-only copies. In addition, the user may enable a feature that sends an email message whenever the owner of the public folder link collection makes additions or changes. In the "gardening" example, if a public folder is found that has a useful collection of links on "gardening", the user can add it to his or her folder tree, and enable email change notifications. Whenever the owner added a new link on gardening, anyone connected to this folder would automatically receive an email notification of the addition.

In contrast to the use of conventional bookmark and favorite features as previously discussed, by using the automatic categorization feature of this system to help in this process, the task of keeping links organized is eased. The user collects bookmarked links in a special folder of "unsorted" links, to be organized at some later time. When a user is ready to organize their "unsorted" folder of links, a "sort" button is be pressed, and the system automatically proposes the best folders in his or her personal link collection to place each unsorted link. To accomplish this the present invention searches the global database, looking to see if other users have put that link in a folder. If matches are found, the folder into which the link was placed is compared with the entire folder collection in the user's personal directory, using the folder matching and categorization process described below. The best matching folder is then proposed as the best location for storing the link. If the link to be sorted is found in more than one folder in the global directory, each folder is matched against the user folder collection, and a consensus proposed file location is computed.

A scoring function scores the strength of the relationship between links. A scoring table is illustrated in Figure 6 which represents the scoring process for the relationship between link 1 in folder A and other links (2-7). A number of factors are used, including the degree of directness of the relationship (the number of folders that are required to connect one link to another), the number of folders that capture the relationship, and the frequency of use of links in the relationship path. The score is calculated as the reciprocal of the length of the shortest folder path between the links. Thus, if the path is 2, the score is ½ = 0.5. If there is no path between the links, the score is 0.

The Figure 5 "Folder Matrix" is an example in which links 1 and 2 have a direct relationship in both folders A and B. These are two different folders that were created in the system in which links 1 and 2 have been grouped together. The more folders in the system that contain these two links, the higher the score assigned to the relationship between links 1 and 2. Alternatively, if few users have grouped links 1 and 2 together in a folder, there is less confidence that the links actually have a meaningful relationship between them, and the score is lower.

The same scoring function is used to assess the degree of indirect relationships, such as the one between links 1 and 4, via link 3 in folders B and D. The more indirect links between 1 and 4, the more likely there is a true relationship present so the higher the score assigned. In addition, the degree of directness is a factor in the scoring function. The more indirect link/folder intersections that must be traversed to relate two links, the smaller the value of the score. The scoring function lowers the score as function of the length of the path. Different well known approaches can be used, including using a linear discount or an exponential decay. The exponential approach accelerates the reduction in score as the path linearly increases in length. Another factor used in the scoring function is to discount paths between links that users are not frequently accessing or that have not recently been accessed and to overweight paths with links that have been frequently and or recently used. For example, consider the connection between links 1 and 4 is via link 3 in folder B and D (see Figure 5 "Folder Matrix"). The score of this connection is reduced if the owners of folders B and D have not recently or frequently used link 3.

The similarity of user profiles (See Figure 4A) is also used in the scoring function. A profile correlation function provides a measure of the degree of the relation between two user profiles. For example, two users with profile information indicating that they are physicians aged 42 and 49 respectively living in the United States are statistically more likely to have link collections that are related than a 42 year old American doctor and a 20 year old farmer from China. Thus, overlapping links found in the folders of more related users are scored higher than the same links in folder of users that are not as related. A consequence of this approach is that matches are proposed which are customized to the profile of the

particular user. The system allows a user to substitute alternative profile information to explore how other people group links together. A young physician with a collection of links on coping with arthritis wishes to the arthritis link collections that older people have collected. The system allows the physician to substitute his or her profile with that of the target user, and to find the best matching links given that target profile.

From Figure 6, a determination is made as to how closely a particular link is related to the links in Folder A. In this example, link 3 is the most closely related, with a high score of 3. Links 4, 5, 6, and 7 are related in progressively decreasing scores, and hence are progressively decreasingly related (see Figure 5 "Folder Matrix"). There, link 3 appears in Folder B, which also contains all the links in Folder A. Thus, Folder B contains all the links in Folder A plus link 3. This indicates that there is potentially a close relationship between link 3 and the links in folder A. If there were more folders in the system that contained the links in folder A, plus link 3, the score for link 3 would increase proportionately to the number of such matches. As another example, link 7 is related to the links in folder A with a lower score of 0.84. The reason that the score is lower is that the connection between link 7 and folder A is indirect, and requires a path through a number of common folders (rather than by a the direct connection between link 3 and folder A). None the less, since there is some connection, the score is greater than 0. When a user is presented with links related to the links in a folder, the order of presentation is based on the score, and larger scores are presented first. There is an arbitrary score cutoff, so that only links where score is sufficiently large are presented. Alternatively, links are presented in decreasing order of score, and the user decides when to stop viewing lower scored links.

In this manner the present invention determines how closely a particular link is related to the links in a given folder. A variation of this approach is used to determine how much a particular folder (which contains a collection of links) is related to another folder. Given an exemplar folder, the system finds closely matching folders in the database. For each folder, the scoring process is applied to determine the score of each link to the exemplar folder. The overall scores for the candidate folder are combined together into a single score for the

entire folder. The average score of the candidate folder is calculated. When this has been done for all folders, the highest overall average scores represent the folders that match most closely to the exemplar folder. Alternatively, a weighted average score is calculated which assigns higher values to folders with large numbers of high scored links and lower values to folders with smaller numbers of related links.

Illustrated in Figure 7, is an example of determined inter-folder relationship. The example table indicates the shortest path via common, shared, links between folders. In the illustration, folder A is connected to folder F by a 3-link path. In other words, the connection between folder A and folder F is indirect. There is no link common to both those folders. To get from A to F one must travel through 2 other folders that share common links. This indicates that folders A and F are probably not very related.

Figure 8 illustrates a flowchart of the matching function of the present invention for matching related bookmarks. In step 800, all appropriate variables are initialized (e.g., *match count, i*). For a given exemplar folder, all links are looped through (Step 802). Related bookmarks are added to *i* from the related table to result set. (Step 804). The match count for related bookmarks is incremented. (Step 806). The total number of matches are recorded. (Step 808). A check is made to see if there are any remaining bookmarks in the folder. (Step 810). If yes, the process loops back to step 804 to process the next bookmarks. If not, the resulting match set is sorted by the match count, (Step 812) and the results are displayed (Step 814).

Shown in Figure 9 is an illustrative flowchart for related link processing under the present invention. After initialization (900), a first bookmark folder is fetched (902). If the end of file is reached (904), then terminate (912). All pairs (i, j) of bookmarks in the instant folder are looped through (906), incrementing the relationship count between the pair (908). This is performed for all the bookmarks in the particular folder. (910). When all bookmarks in one folder have been processed, the program goes to process the next folder (902) until finished.

An exemplary Java implementation of the present invention including the matching function of Figure 8 and related link processing of Figure 9

is attached as Appendix I.

The result of this process is a list of links and scores as discussed with respect to Figure 6. The higher the score, the more closely related the link was to related clusters of the links in the folder database. An example output from the algorithm of Appendix I is included as Appendix II.

In summation, the system of the present invention thus helps to overcome deficiencies of conventional automatic web search engines discussed above, as matching is done by a consensus of users associating links in folders. Even if one user put the book store link along with a collection of links on the United States Government, the weighting function used rejects that association (unless, of course, many people made that association in their link folders, which might mean that there was some legitimate connection between that book store and e.g., the White House).

The automatic folder-based categorization and private user link directories are used to help in the creation of a comprehensive web directory. As previously discussed, users file links in their own personal folder hierarchy. The matching and categorization process uses this information to find related links and folders. This basic mechanism is extended to form a mechanism for creating a global web link directory. The process works as follows. Effectively, all of the private user folders are intersected to form a master folder and link directory. Where, e.g., a fairly complete master link directory already exists, as users add new links to their private folders, the folder matching process is used to determine which master folder is the best place in which it should be stored. This is accomplished by searching all private user folders that contain the link. As users add the link to their own private collection of links, the system "learns" which other links are most closely related on the basis of the other links in the same folder. The more private user folders that contain this link, the more data there is available for finding related links. With this information, the process searches the master link directory to find the folder that best matches the private user folders that contain the link. The link is added to the one or more master folders that contain the closest related links.

This is the process for adding links to a folder in the master link directory. The basic matching process also is used to suggest the folder hierarchy

ordering of the master link directory. That is, it determines the folders into which a subfolder is place. The result is a tree like structure for the storage of links. Where, for example, a folder called "gardening" contains subfolders called "vegetable" and "flowers", the problem becomes, how can the master link directory be organized this way, based on the private organization structures of individual users? The match is used to determine the user link folders that best match a master link folder. The matching is performed across all users, and then the user private link folders are examined. Across many user link collections are found that, e.g., match the flower link folder and in most case the vegetable link folder, that are contained in a common parent folder. Thus, the system scans users private link folders to learn this taxonomy.

To provide incentive to users of the link categorization system to create useful collections of public links, a payment incentive scheme can be implemented at the central server in an alternative embodiment. As previously described the system tracks the number of users that reference a public link folder. With this reference count mechanism, along with access statistics (how frequently people are clicking on links in a public folder), formulas have been developed to rank the popularity of a public folder. A simple approach is to rank the folders based on the number of read-only references. The more read-only references from other users, the higher the ranking. The highest ranking folder creators could receive payment for their contributions made to the overall link collection. This payment scheme would also be useful for promoting the service, as people would spread the word about their own personal link collection, to increase the amount of payment that they would receive for their collection. A modification to the simple reference counting scheme for determining payment is to have a blending function that included both total number of references and the number of clicks on a collection over a period of time. A linear combination of the two parameters is used, with the scale factors selected based on the total payment funds available.

The development of "communities" on the web is an area that receives much interest. Web communities are groups of people that are brought together to the same web site that share a common set of interests. Various approaches are in use to achieve this, including the creation of discussion groups

around a specific topic. As an example, a site like Etrade has discussion groups on individual stocks and investment ideas. These methods suffer from a number of shortcomings, including being overly specific to a set of topics chosen by the community builder. None allow the users to effectively build a community around a more abstract set of interests. The use of the categorization and matching function can be used to overcome this. These functions can be used to find people with similar link collections, which is a strong indicator that these people share the interests reflected by the content of their similar link collections. As an example, a user creates a folder of links on "gardening". The system finds other users that have such a link collection, and invites them to join a dynamically created discussion group of all users with similar gardening folders. As new people in the overall community of users create folders that match the gardening folder, they too are invited to join the group. This process for inviting people to join the community described overcomes privacy concerns. Users are asked to join the community. Until they join, other users are not made aware of their existence. In a similar manner, discussion groups are optionally supported on any public folder. If a public folder owner chooses to do so, he or she may enable the discussion feature. This allows anyone accessing the public folder to post a message that will be displayed along with the public folder links. Since it is likely that the group of people accessing the public folder share the interests reflected by the content referenced by the links in the public folder, this facilitates the creation of a community that evolves around that specific interest.

Alternative uses in varying embodiments include relating and scoring links for targeting web advertising. To achieve this, a potential advertiser creates a collection of links that is believed to be representative of the product or service that is to be marketed. For example, if an advertiser were selling gardening products, a collection of gardening links is gathered and placed in an exemplar link folder. Using the folder matching and categorization process, matching user link folders are found. A link to the advertiser's web site is then automatically inserted into the user's link collection, and appears whenever the user enters that folder. Multiple advertiser links may be inserted into the same folder, permitting all garden advertisers to insert a reference to the web site's URL in all gardening-related web

folders in the system. The same approach is used to display a banner ad at the top of a page whenever a user enters a folder that matches an exemplar from an advertiser.

Using CGI scripts (a known process used to generate dynamic web pages), a user can insert a dynamic hot list on any web page thereby creating embedded link lists. An exemplary Java implementation of the embedded link list feature is attached as Appendix III. An access module that runs on a remote computer makes a database connection to a central server. This server returns the contents of a link folder to the remote application. The folder is then displayed on the web page as a hot list. This approach makes the hot list dynamic. As additions or changes are made to the folder, the next time the web page is accessed, the new additions or changes are displayed. Using the matching and categorization function, the hot list is supplemented by other similar links, and includes not only the links in the particular folder, but closely related ones as well.

The present invention has been illustrated and described with respect to specific embodiments and applications thereof. To facilitate discussion of the present invention, a preferred embodiment is assumed, however, that the above-described embodiments are merely illustrative of the principles of the invention and are not intended to be exclusive embodiments thereof. It should be understood by one skilled in the art that alternative embodiments drawn to variations in the enumerated embodiments and teachings disclosed herein can be derived and implemented to realize the various benefits of the present invention.

It should further be understood that the foregoing and many various modifications, omissions and additions may be devised by one skilled in the art without departing from the spirit and scope of the invention. It is therefore intended that the present invention is not limited to the disclosed embodiments but should be defined in accordance with the claims which follow.

APPENDIX I: Sample Java Source Code

```
// Java class function to analyze a series of link
// directors
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

import Folder;
import Directory;
import Permutation;

class Analyze {
  Directory _dir;
  Folder _f;
  double[] _results;

  // Constructor that creates an instance of the analyze
  // class.  We pass in the directory tree that contains
  // all link folders in the system and a folder to be
  // matched against the directory.
  public Analyze(Directory dir, Folder f) {
    _dir = dir;
    _f = f;
    _results = new double[1000];
    int i;
    for (i=0; i<1000; i++)
      _results[i] = 0.0;
  };

  // Compute all combinations of the folder given in the
  // analyze contructor and run the matching algorithm on
  // each.  We take all combinations of m links taken n
  // at a time, where m is the number of links in the
  // exemplar folder and n ranges from m to 1.
  public Folder permute(double thresh, int level) {
    int[] x = _f.toArray();
    int n = _f.count();
    int i;

    // This bootstraps the combination process by calling
    // the combination helper routine ones for each m
    // (where m is the number of items being considered
    // in the combination, as described above.
    for (i=n-1; i>0; i--) {
      int[] p = new int[i];
      doPermute(p, x, n, i, i);
    }

    Folder folder = new Folder();

    // We build a folder here with the summary results
    // from all matches that we found by running the
    // above permutation.  Note that we only include
    // matches that exceed a given score threshold, as
    // specified in an argument to this routine.  We
    // weight the score by the "level", where level is
    // the number of times the routine was reinvoked on
```

APPENDIX I: Sample Java Source Code

```
    // its own results.  This is how indirect
    // relationships are found.
    for (i=0; i<1000; i++) {
      double v = _results[i] / (double)level;
      if ((v > 0) && !_f.contains(i)) {
        System.out.println(i + " ---> " + v);
        if (v > thresh)
          folder.addLink(i);
      }
    }

    return folder;
  };

  // This method does the work of the matching process.
  // The actual combinations m at a time are computed
  // here, recursively.  The cumulative result score is
  // also computed.
  //
  // Arguments to the routine:
  // p: accumulated permutations
  // x: input array
  // n: length of input array
  // m: remaining count to permute
  // c: total length of permutation
  private void doPermute(int[] p, int[] x, int n,
              int m, int c)
  {
    int i, j;

    // Create a copy of the input link list omitting one
    // link at a time.  This is done to compute the
    // combinations of links.
    for (i=0; i<n; i++) {
      int[] xCopy = new int[n-1];
      for (j=0; j<n-1; j++) {
        if (j<i)
          xCopy[j] = x[j];
        else
          xCopy[j] = x[j+1];
      }

      // Build the new combination of links to be
      // considered.
      p[m-1] = x[i];

      // If the length of the remaining links to be added
      // to the combination is greater than one, invoke
      // the routine recursively to add the remaining
      // links to the combination.  If not, we've got a
      // combination and are ready to compute a score.
      if (m > 1) {
        doPermute(p, xCopy, n-1, m-1, c);
      } else {
        // Build a link folder from the combination.
        Folder f = new Folder(p, c);

        // Match this folder against the directory tree,
        // looking for all folders that contain this
```

APPENDIX I: Sample Java Source Code

```
        // exact set of links.
        Directory.Result rc = _dir.findSimilar(f);

       // Create a score weighting factor. Here, c is the
      // total length of the permutation and _f.count() is
      // the total length of the initial exemplar
      // provided.  Thus, the more links from the original
      // exemplar provided that are in this combination,
      // the greater the score.
        double factor = (double)c/(double)_f.count();

        // Now scan all matches and record the scores for
        // each link. Only keep the highest score, since
        // the link could be find in more than one of the
        // combination passes through this routine.
        int k;
        for (k=0; k<1000; k++) {
          double v = (rc.folders[k]*factor)/rc.total;
          if (v > _results[k])
            _results[k] = v;
        }
      }
    }
  };

  // Print a summary of the results.
  public int printStats(Folder folder, int results[],
     int total, double factor)
  {
    int i;
    for (i=0; i<1000; i++) {
      if (results[i] > 0)
        System.out.println("Found '" + i + "' " +
          results[i] + " times (" +
          ((float)results[i]*factor)/(float)total + ")");
    }
    return total;
  };

  // Helper function to print an array.
  public void printArray(int[] array, int count) {
    int i;
    for (i=0; i<count; i++)
      System.out.print(array[i] + " ");
    System.out.println();
  };
}
//
// Java functions to manipulate a system wide directory
// of link folders.
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

import Folder;
```

APPENDIX I: Sample Java Source Code

```
class Directory {
  private Folder[] _folders;
  private int _allocated;
  private int _inUse;

  // Create an empty directory (limitted to 1000 link folders for now).
  public Directory() {
    _folders = new Folder[1000];
    _allocated = 1000;
    _inUse = 0;
  };

  // Add a folder from a string to the system wide folder
  // directory.
  public int addFolder(String s) {
    Folder f = new Folder();
    f.addLinks(s);
    return addFolder(f);
  };

  // Add a folder from a folder class to the system wide
  // folder directory.
  public int addFolder(Folder f) {
    _folders[_inUse] = f;
    return _inUse++;
  };

  // Count the number of times all links in an exemplar
  // folder match against folders in the system wide
  // directory.
  public int match(Folder exemplar) {
    int count = 0;
    int i;

    for (i=0; i<_inUse; i++) {
      if (_folders[i].match(exemplar))
        count++;
    }

    return count;
  };

  // Helper class that is the return value of the next
  // function.
  class Result {
    int[] folders;
    int total;
  };

  // This function takes an exemplar folder as an input,
  // and return an array that counts how many times each
  // link in the exemplar appears in the directory.  For
  // example, folders[i] of the result contains the count
  // of how many times link i matched against all the
  // links in the directory.
  public Result findSimilar(Folder exemplar) {
    int results[] = new int[1000];
    int matches = 0;
    int i;
```

APPENDIX I: Sample Java Source Code

```
    // Initalize the return result counts to 0.
    for (i=0; i<1000; i++)
      results[i] = 0;

    // For each folder in the system, see if we have
    // matches against the exemplar.  If we do, add one
    // to the count for a match for a link each time a
    // match has been found.
    for (i=0; i<_inUse; i++) {
      if (_folders[i].match(exemplar)) {
        matches++;
        Enumeration elems = _folders[i].keys();
        while (elems.hasMoreElements()) {
          Integer link = (Integer)elems.nextElement();
          if (!exemplar.contains(link))
            results[link.intValue()]++;
        }
      }
    }

    // Build and return the results.
    Result rc = new Result();
    rc.folders = results;
    rc.total = matches;
    return rc;
  };
}
```

APPENDIX I: Sample Java Source Code

```
//
// Java functions to manipulate a link folder.
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

class Folder {
  private Hashtable _folder;

  // Constructor to create an empty link folder.
  public Folder() {
    _folder = new Hashtable();
  };

  // Constructor to create a link folder from a string
  // listing the links in the folder.
  public Folder(String linkString) {
    _folder = new Hashtable();
    addLinks(linkString);
  };

  // Constructor to create a link folder from an array,
  // where only the links that have a score greater than
  // the given threshold are added to the link folder.
  public Folder(int[] linkArray, int divisor,
    double thresh)
  {
    int i;
    _folder = new Hashtable();
    for (i=0; i<1000; i++) {
      if ((double)linkArray[i]/(double)divisor > thresh)
        _folder.put(new Integer((int)i), "");
    }
  };

  // Constructor to create a link folder from an array.
  public Folder(int[] array, int count) {
    int i;
    _folder = new Hashtable();
    for (i=0; i<count; i++) {
      _folder.put(new Integer((int)array[i]), "");
    }
  };

  // Return the links in the folder.
  public Enumeration keys() {
    return _folder.keys();
  };

  // Add a bunch of links in string format to a folder.
  public int addLinks(String linkString) {
    Reader in;
    in = new StringReader(linkString);
    StreamTokenizer parser = new StreamTokenizer(in);

    try {
```

APPENDIX I: Sample Java Source Code

```
    while (parser.nextToken() !=
                              streamTokenizer.TT_EOF)
    {
      switch (parser.ttype) {
      case StreamTokenizer.TT_EOL:
        break;
      case StreamTokenizer.TT_NUMBER:
        _folder.put(new Integer((int)parser.nval), "");
        break;
      }
    }
  } catch (IOException e) {
    return -1;
  }

  return 0;
};

// Method to add a link to the folder.
public void addLink(int link) {
  _folder.put(new Integer((int)link), "");
}

// Determine if the folder contains a given link.
public boolean contains(int link) {
  return _folder.containsKey(new Integer(link));
};

// Same as above for Integer type.
public boolean contains(Integer link) {
  return _folder.containsKey(link);
};

// Determine if an example folder matches the given
// folder.
public boolean match(Folder exemplar) {
  Enumeration elems = exemplar._folder.keys();
  boolean found = true;

  // Iterate through all the links in the exemplar and
  // make sure each and every one is in the folder.
  while (elems.hasMoreElements()) {
    Integer link = (Integer)elems.nextElement();
    if (!contains(link.intValue())) {
      found = false;
      break;
    }
  }

  // Return true if we have a match.
  return found;
};

// Determine if an example folder matches the given
// folder.
public void print() {
  Enumeration elems = _folder.keys();
  while (elems.hasMoreElements()) {
    Integer link = (Integer)elems.nextElement();
```

APPENDIX I: Sample Java Source Code

```
      System.out.print(link + " ");
    }
    System.out.println();
  };

  // Convert to an array.
  public int[] toArray() {
    int[] array = new int[_folder.size()];
    Enumeration elems = _folder.keys();
    int i = 0;
    while (elems.hasMoreElements()) {
      Integer link = (Integer)elems.nextElement();
      array[i++] = link.intValue();
    }
    return array;
  };

  // Return the number of links in the folder.
  public int count() {
    return _folder.size();
  };
}
```

APPENDIX I: Sample Java Source Code

```
//
// Main test harness for the link matching algorithms.
// Author: David Siegel
//

import java.util.*;
import java.io.*;
import java.lang.*;

import Folder;
import Directory;
import Analyze;

class Main {
  public static void main(String argv[]) {
    Directory dir = new Directory();

    // Add a bunch of folders to the system directory.
    // Each number represents a link number.  Each call
    // to addFolder adds a new folder.  For the purposes
    // of this test we do not track the folder directory
    // structure or the folder owner.
    dir.addFolder("1 2 3 5");
    dir.addFolder("1 2 3 5");
    dir.addFolder("1 2 3 5");
    dir.addFolder("1 2 3 7");
    dir.addFolder("2 2 3 7");
    dir.addFolder("2 2 3 9");
    dir.addFolder("2 2 3 5");
    dir.addFolder("1 5 9 10");
    dir.addFolder("1 5 9 10");
    dir.addFolder("1 5 9 11");
    dir.addFolder("1 5 9 11");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("1 5 9 12");
    dir.addFolder("10 11 12 44");
    dir.addFolder("12 15 19 23");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");
    dir.addFolder("12 15 19 22");

    // Perform the matching process with the specified
    // exemplar folder.  Thus, the input to the matching
    // process is the system wide folder directory that
    // was created above, plus an exemplar folder.  The
    // goal is to find links that are related to the
    // input exemplary.
    System.out.println("Level 1:");
    Analyze study = new Analyze(dir,
                          new Folder("1 2 3 4 5 9"));
```

APPENDIX I: Sample Java Source Code

```
    Folder f = study.permute(0.01, 1);

    // We now reinvoke the matching algorithm on the
    // results of the first match.  This is in essence a
    // level 2 match which finds indirectly related links
    // to the initially given exemplar.  This process
    // could be repeated to find level 3, 4, etc.
    // relationships, though after a while the
    // relationships become too distant to be meaningful.
    System.out.println("Level 2:");
    study = new Analyze(dir, f);
    study.permute(0.01, 2);
  };
}
```

APPENDIX I: Sample Java Source Code

```
/*    $Id: RelatedUrl.java,v 1.4 Exp $    */
// Author: David Siegel
//
package BlinkUtil;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.Date;
import java.text.*;
import java.sql.*;
import java.sql.SQLException;

public class RelatedUrl extends Url {

  private int _matchCount;

  public void finalize() throws Throwable {
    super.finalize();
  }

  public RelatedUrl(Session session, ResultSet rs) throws SQLException {
    super(session, rs);
    _matchCount = rs.getInt("match_count");
  }

  public int matchCount() {
    return _matchCount;
  }

  public void addMatchCount(RelatedUrl rurl) {
    _matchCount += rurl._matchCount;
  }
}
```

APPENDIX I: Sample Java Source Code

```
/*    $Id: RelatedUrlDir.java,v 1.7 Exp $ */
// Author: David Siegel
//
package BlinkUtil;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.Date;
import java.text.*;
import java.sql.*;
import java.sql.SQLException;

public class RelatedUrlDir {

  Session _session;
  Hashtable _ht;
  RelatedUrl[] _rurls;

  public void finalize() throws Throwable {
    super.finalize();
    _session = null;
    _ht = null;
    _rurls = null;
  }

  public RelatedUrlDir(Session session) {
    _session = session;
    _ht = null;
    _rurls = null;
  }

  public Session session() {
    return _session;
  }

  public RelatedUrl[] items() {
    return _rurls;
  }

  public int size() {
    if (_rurls == null)
      return 0;
    else
      return _rurls.length;
  }

  public void add(ResultSet rs) throws SQLException {
    RelatedUrl next = new RelatedUrl(_session, rs);
    RelatedUrl rurl = (RelatedUrl)_ht.get(new Integer(next.urlId()));
    if (rurl == null)
      _ht.put(new Integer(next.urlId()), next);
    else
      rurl.addMatchCount(next);
  }

  public void update(int dirId) throws SQLException, FatalException {
    _ht = new Hashtable();
```

APPENDIX I: Sample Java Source Code

```
    SQL.RelatedUrlDir(this, dirId);

    if (_ht.size() > 0) {
      _rurls = new RelatedUrl[_ht.size()];
      Enumeration elems = _ht.elements();
      int i = 0;
      while (elems.hasMoreElements()) {
      RelatedUrl rurl = (RelatedUrl)elems.nextElement();
      _rurls[i++] = rurl;
      }
      sort();
    } else
      _rurls = null;
  }

  private void sort() {
    int i, j;
    for (i = 0; i < _rurls.length-1; i++) {
      for (j = i+1; j < _rurls.length; j++) {
      if (_rurls[i].matchCount() < _rurls[j].matchCount()) {
        RelatedUrl swap = _rurls[i];
        _rurls[i] = _rurls[j];
        _rurls[j] = swap;
      }
      }
    }
  }
}
```

APPENDIX I: Sample Java Source Code

```
/*$Id: RelatedUrlGenerate.java,v 1.5 Exp $       */
// Author: David Siegel
//
package BlinkUtil;

import java.io.*;
import java.lang.*;
import java.util.*;
import java.util.Date;
import java.text.*;
import java.sql.*;
import java.sql.SQLException;

public class RelatedUrlGenerate {

  class Item {
    private int _urlId;
    private int _count;

    public Item(int urlId) {
      _urlId = urlId;
      _count = 0;
    }

    public int urlId() { return _urlId; }
    public int count() { return _count; }
    public void increment() { _count++; }
  }

  Hashtable _ht;

  public RelatedUrlGenerate() {
    _ht = null;
  }

  public void add(ResultSet rs) throws SQLException {
    int matchUrlId = rs.getInt("url_id");
    Item item = (Item)_ht.get(new Integer(matchUrlId));
    if (item == null) {
      item = new Item(matchUrlId);
      _ht.put(new Integer(matchUrlId), item);
    }
    item.increment();
  }
```

APPENDIX I: Sample Java Source Code

```
  public void update(int urlId) throws SQLException, FatalException {
    _ht = new Hashtable();

    SQL.RelatedUrlFind(this, urlId);

    Enumeration elems = _ht.elements();
    while (elems.hasMoreElements()) {
      Item item = (Item)elems.nextElement();
      SQL.RelatedUrlUpdate(urlId, item.urlId(), item.count());
    }
  }
}
```

APPENDIX II: Sample Output

```
Level 1:
7 ---> 0.125
10 ---> 0.1
11 ---> 0.1
12 ---> 0.3
Level 2:
1 ---> 0.08333333333333333
2 ---> 0.125
3 ---> 0.125
5 ---> 0.08333333333333333
9 ---> 0.08333333333333333
15 ---> 0.0703125
19 ---> 0.0703125
22 ---> 0.0625
23 ---> 0.0078125
44 ---> 0.375
```

APPENDIX III: Sample Java Source Code

```
/*    $Id: Embed.java,v 1.7 Exp $    */
// Author: David Siegel
//
import com.blink.blink.Config;
import java.io.*;
import java.lang.*;
import java.util.*;
import java.net.MalformedURLException;
import java.sql.SQLException;
import javax.servlet.*;
import javax.servlet.http.*;

import BlinkUtil.*;

public class Embed extends HttpServlet {

  private static Session _session;
  private State _state;
  private int _count = 0;

  public void init(ServletConfig servletConfig) throws ServletException {
    super.init(servletConfig);
    try {
      Config.load();
    } catch (IOException e) {
      throw new ServletException("Can't load configuration data:" +
          e.getMessage());
    }
    Log.init();
    _state = State.getState();
    _session = new Session(null, null, null, null);
    Config config = Config.getConfig();
    _session.dispatch(config.get("html.url.server.root") +
      config.get("html.url.run"));
  }

  private String fmt(String html) {
    html = Page.quoteQuote(html);
    return "document.write(\"" + html + "\");\n";
  }

  public void service(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException {
    ServletOutputStream out = res.getOutputStream();
    res.setContentType("text/html");

    String cmd = req.getParameter("cmd");

    if (cmd.equals("gen")) {
      String dirIdStr = req.getParameter("dir");
      String options = req.getParameter("opt");
      try {
      EmbedLinks e = new EmbedLinks(_session);
      String str = e.html(dirIdStr, options);
      if (Log.Debug) Log.log(Log.Dispatch, fmt(str));
      out.println(fmt(str));
```

APPENDIX III: Sample Java Source Code

```
      } catch (Exception e) {
      out.println(fmt("Folder not available"));
      if (Log.Syslog) Log.log(Log.Embed, "Embed Folder(" + dirIdStr +
        ", " + options + ") got:" + e.getMessage());
      }
    } else if (cmd.equals("hit")) {
      String urlIdStr = req.getParameter("url");
      String dirIdStr = req.getParameter("dir");
      try {
      int urlId = Page.getId(urlIdStr, "url");
      int dirId = Page.getId(dirIdStr, "dir");
      Url url = _session.getUrl(urlId, dirId);
      url.hit(dirId);
      res.sendRedirect(url.href());
      } catch (Exception e) {
      out.println(fmt("Cannot launch URL"));
      if (Log.Syslog) Log.log(Log.Embed, "Embed hit(" + urlIdStr + "," +
        dirIdStr + ") got:" + e.getMessage());
      }
    }
  }
}
```

Claims

I claim:

1. A method for managing World Wide Web links, comprising the steps of:

providing at least one list of web links from each of a plurality of users;

analyzing the lists to determine a pair of lists from different users of the plurality of users that are closely correlated;

synthesizing a link list that collects the links from the correlated pair, and presenting the synthesized link list in a folder for shared use among the plurality of users.

2. The method of claim 1 further comprising the steps of providing a master link directory and a user profile and analyzing the master link directory to form collections of links of relevant content that closely relates to the user profile.

3. The method of claim 1 further comprising the steps of determining folder grouping of links by other users and categorizing the links based on how other users group the links into folders.

4. The method of claim 1 further comprising generating a dynamic link collection based on a set of exemplar links on a web page.

5. The method of claim 1 further comprising building web communities by grouping users together that have similar collections of links.

6. The method of claim 1 further comprising inserting particular links into the relevant link folder.

7. The method of claim 1 further comprising locating public link folders by matching against an exemplar link or link collection.

8. The method of claim 2 wherein user preferences can be added to the user profile to weight matches according to how closely related the users are, or display links that are interesting to other user profiles having the same preferences.

9. The method of claim 7 further comprising the step of notifying the users of updates when links are added to public folders.

10. The method of claim 9 wherein said step of notifying includes emailed notification of said updates.

11. The method of claim 1 further comprising the step of removing dead links from a user link collection.

12. The method of claim 2 further comprising the steps of generating master link folders based on the collective organization of private link folders, by combining together the grouping information of each individual user.

13. The method of claim 1 wherein the step of analyzing includes organizing a collection of link folders into a taxonomy or hierarchy of link folders that uses private user link folder hierarchies and that finds the best representative hierarchy for a master link tree.

14. The method of claim 13 wherein the step of analyzing further includes suggesting user hierarchies or folder organizations by comparing a user link folder against the other users.

15. The method of claim 1 further comprising the steps of providing payment to users with popular read-only links based on the number of relation hits determined for that link.

16. The method of claim 8 further comprising the step of having the user rate the appropriateness of discriminated matches, which is then incorporated into the weighting function to further refine future matches.

17. The method of claim 1 further comprising the step of track of the total number of visits to a link for each individual user of said plurality of users and for all of said plurality of users.

18. The method of claim 8 further comprising the step of searching said synthesized link list by using (1) a history of a particular link, which history can include the last date the link was referenced, the number of times the link has been accessed by a particular user of said plurality of users and by all of said plurality of users, a user-provided rating of the link, and/or descriptive textual information, and (2) information obtained from other users who have the same link to their respective link list.

19. The method of claim 2 further comprising creating a folder of unsorted links for links to be sorted at a later time.

20. The method of claim 19 further comprising the step of automated sorting of said folder of unsorted links by proposing the most relevant folders in said user's link collection in which to place each unsorted link.

21. The method of claim 2 further comprising the step of determining other users who have similar link collections to that of a particular user, and inviting said other users to join a dynamically created discussion group of all such users with similar link collections.

22. The method of claim 6 wherein said step of inserting particular links into said relevant link folder includes;

creating an exemplar collection of links representative of a product or service that is to be marketed;

matching user link folders are found;

inserting into the user's link collection a particular link; and

displaying said inserted link whenever the user enters that folder that matches an exemplar.

23. The method of claim 22 wherein a banner ad is displayed at the top of a page whenever the user enters that folder.

24. The method of claim 1 further comprising the step of creating embedded link lists.

25. A system for managing World Wide Web links, comprising:

a plurality of remote individual web users;

centralized data storage for storing at least one list of web links from each of the plurality of users;

a central server comprising means for discriminating a correlated pair of lists from different users of the plurality of users and for synthesizing a link list that collects the links from the correlated pair, and means for storing the synthesized link list in a folder in the centralized data storage for shared access use among the plurality of users.

26. The system of claim 25 wherein the centralized data storage hosts user profiles for the individual users.

27. The system of claim 26 wherein the stored user profiles include user preferences and the central server further includes means for weighting

matches according to how closely related the users are, or display links that are interesting to other user profiles having the same preferences.

28. The system of claim 25 wherein the central server further comprises means for notifying the users of updates when links are added from the synthesized link lists.

29. The system of claim 25 wherein the central server further comprises means for facilitating provision of payment to users with popular read-only links based on the number of relation hits determined for that link.

30. The system of claim 25 wherein the central server further comprises means for facilitating user rating of the appropriateness of discriminated matches.

31. The system of claim 25 further comprising means for inserting particular links into the relevant link folder.

32. The system of claim 25 further comprising means for locating public link folders by matching against an exemplar link or link collection.

33. The system of claim 32 further comprising means for notifying the users of updates when links are added to public link folders.

34. The system of claim 33 wherein said means for notifying includes email notification.

35. The system of claim 25 further comprising means for providing payment to users with popular read-only links based on the number of relation hits determined for that link.

36. The system of claim 27 further comprising means for allowing the user to rate the appropriateness of discriminated matches and incorporating the rating into the weighting function to further refine future matches.

37. The system of claim 25 further comprising means for tracking the total number of visits to a link for each individual user of said plurality of users and for all of said plurality of users.

38. The system of claim 27 further comprising means for searching said synthesized link list by using (1) a history of a particular link, which history can include the last date the link was referenced, the number of times the link has been accessed by a particular user of said plurality of users and by all of

said plurality of users, a user-provided rating of the link, and/or descriptive textual information, and (2) information obtained from other users who have the same link to their respective link list.

39. The system of claim 25 further comprising:

means for providing a master link directory and a user profile and for analyzing the master link directory to form collections of links of relevant content that closely relates to the user profile;

means for creating a folder of unsorted links for links to be sorted at a later time; and,

means for automated sorting of said folder of unsorted links by proposing the most relevant folders in said user's link collection in which to place each unsorted link.

40. The system of claim 31 wherein said means for inserting particular links into said relevant link folder includes;

means for creating an exemplar collection of links representative of a product or service that is to be marketed;

means for matching user link folders are found;

means for inserting into the user's link collection a particular link; and

means for displaying said inserted link whenever the user enters that folder that matches an exemplar.

41. The system of claim 25 further comprising means for creating embedded link lists.

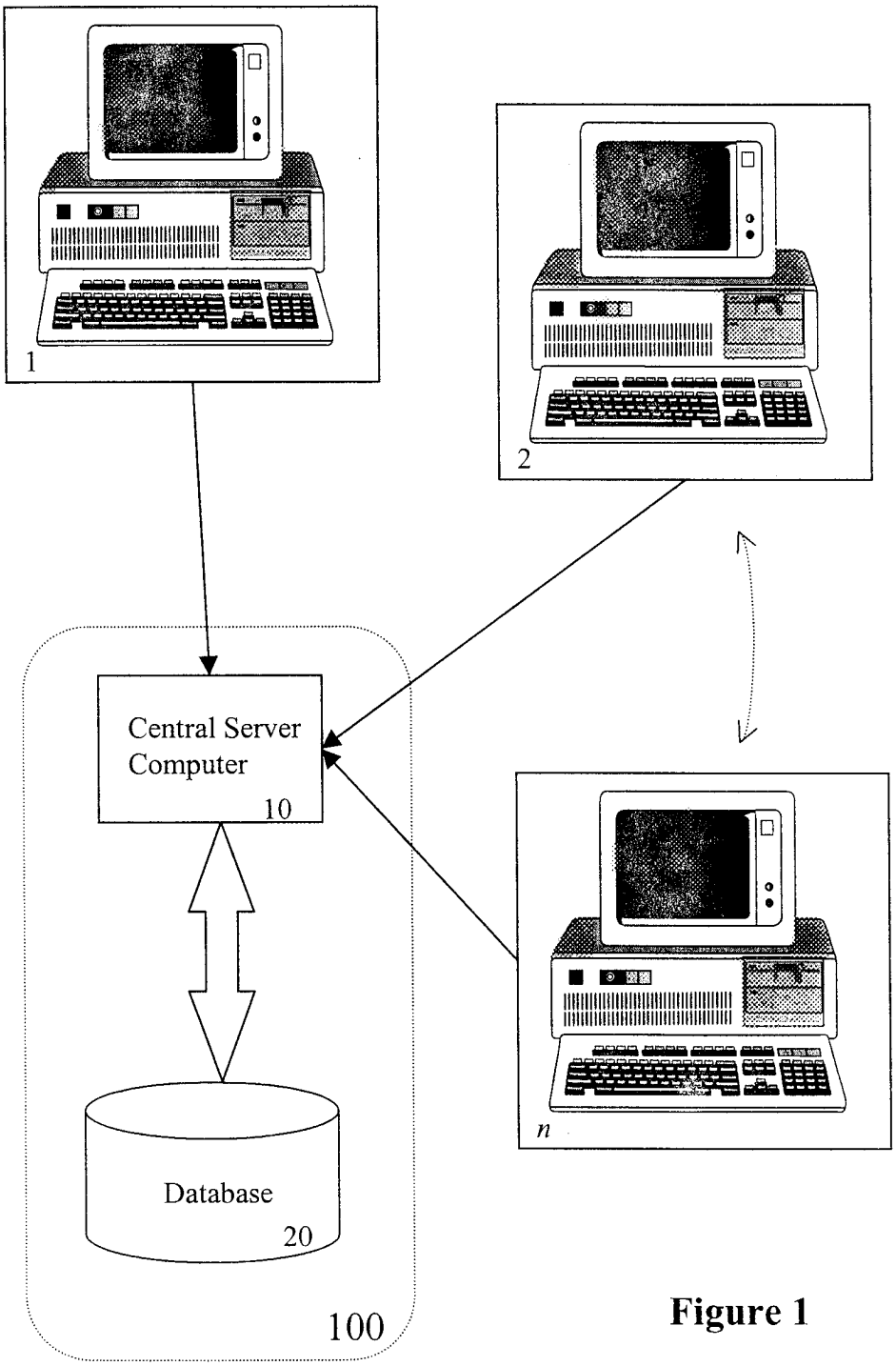


Figure 1

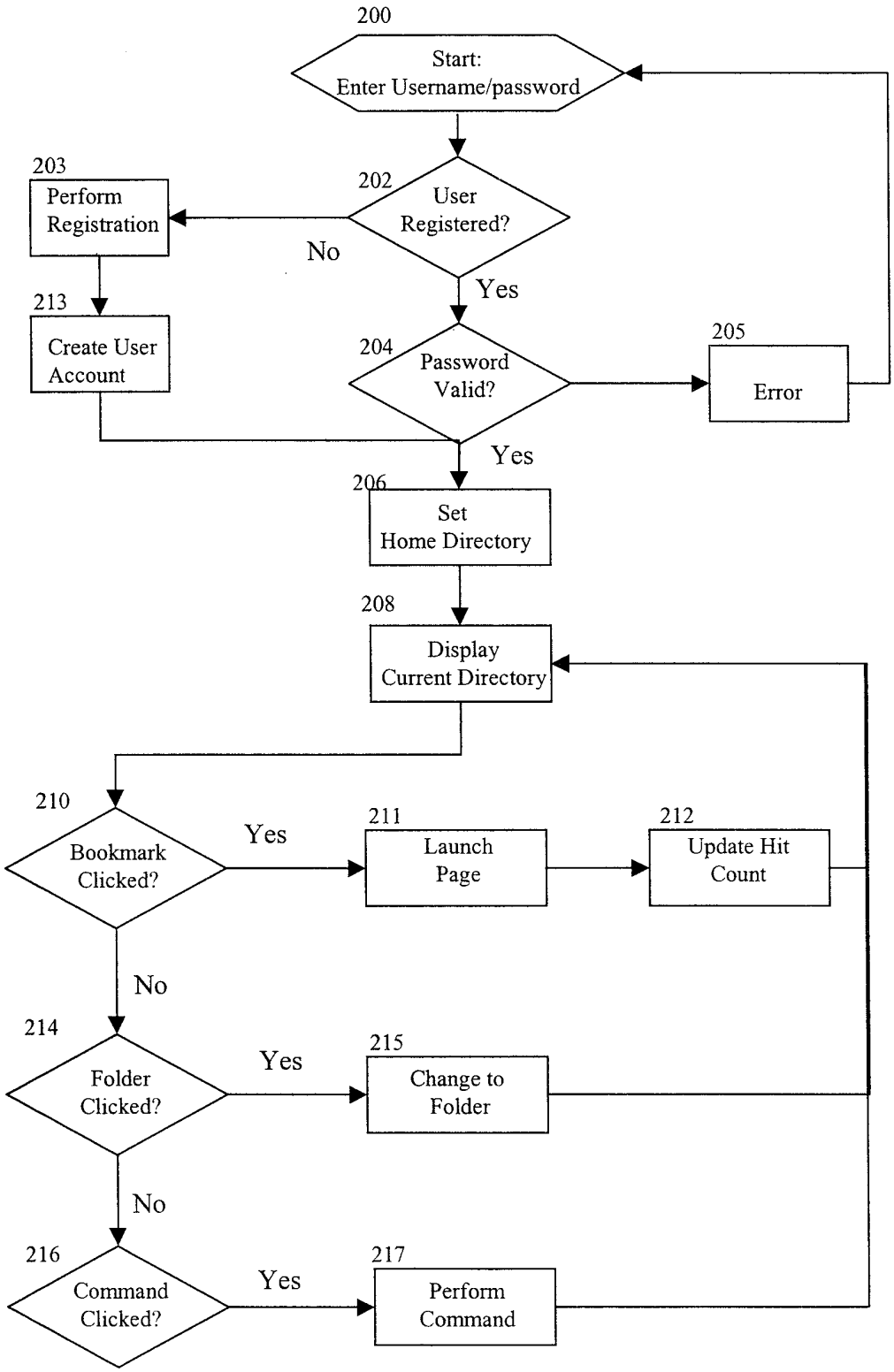

200
Start:
Enter Username/password
202
User
Registered?
No
203
Perform
Registration
213
Create User
Account
Yes
204
Password
Valid?
205
Error
Yes
206
Set
Home Directory
208
Display
Current Directory
210
Bookmark
Clicked?
Yes
211
Launch
Page
212
Update Hit
Count
No
214
Folder
Clicked?
Yes
215
Change to
Folder
No
216
Command
Clicked?
Yes
217
Perform
Command

Figure 2

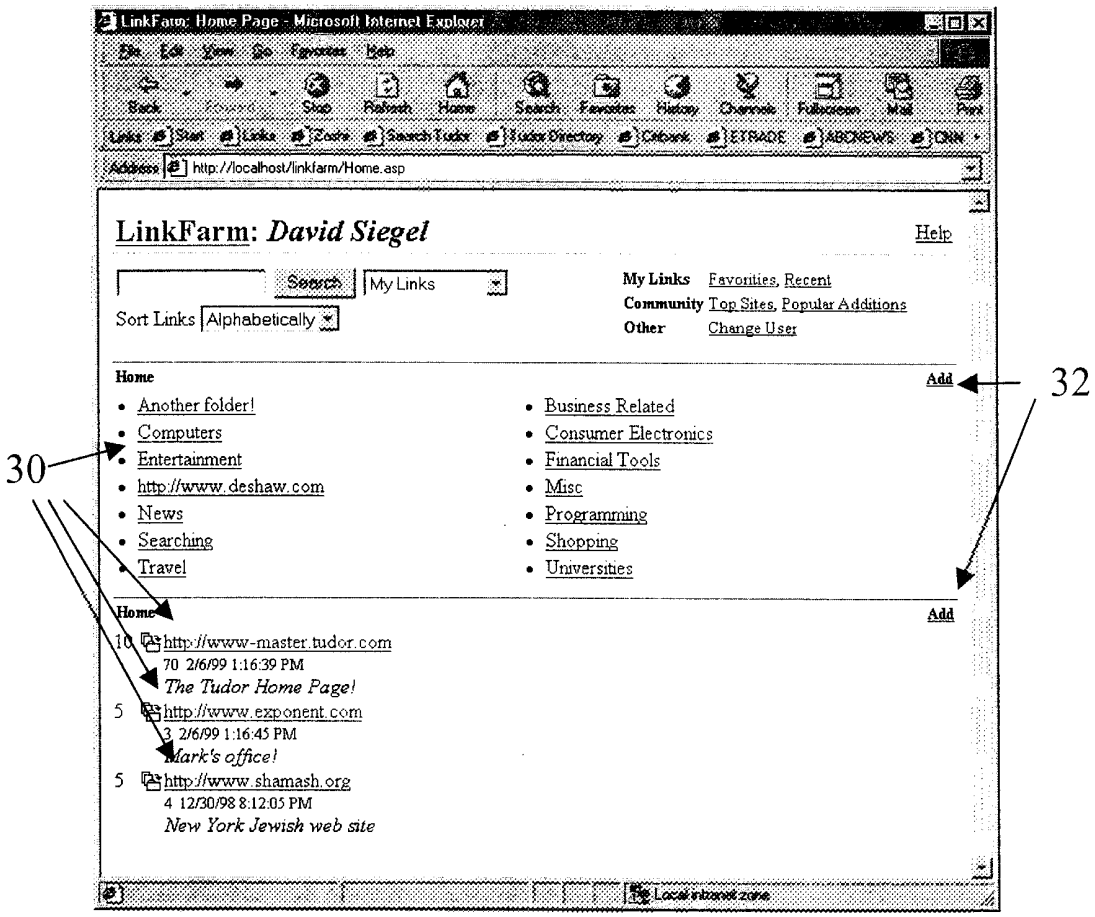

LinkFarm: Home Page - Microsoft Internet Explorer
http://localhost/linkfarm/Home.asp
LinkFarm: David Siegel
Help
Search
My Links
Sort Links
Alphabetically
My Links Favorites, Recent
Community Top Sites, Popular Additions
Other Change User
Home
Add
Another folder!
Computers
Entertainment
http://www.deshaw.com
News
Searching
Travel
Business Related
Consumer Electronics
Financial Tools
Misc
Programming
Shopping
Universities
Home
Add
10 http://www-master.tudor.com
70 2/6/99 1:16:39 PM
The Tudor Home Page!
5 http://www.exponent.com
3 2/6/99 1:16:45 PM
Mark's office!
5 http://www.shamash.org
4 12/30/98 8:12:05 PM
New York Jewish web site
30
32

Figure 3

FIELD	DESCRIPTION
UserId	Unique user identification number
UserName	Unique user name
First Name	First name
Last Name	Last name
Occupation	Job occupation code
Age	Age (or age range)
Income	Income

Figure 4A

FIELD	DESCRIPTION
FolderID	Unique folder identification number
UserID	User ID of the folder owner

Figure 4B

	A	B	C	D	E	F
1	X	X				
2	X	X	X			
3		X	X	X		
4				X	X	
5				X	X	X
6					X	X
7						X

Figure 5

	B	C	D	E	F	Total
2						
3	1	1	0.5	0.5	0	3
4	0	1	0.5	0.5	0	2
5	0	0	0.5	0.5	0	1
6	0	0	0.5	0.5	0.34	1.34
7	0	0	0	0.5	0.34	0.84

Figure 6

	A	B	C	D	E	F
A	0					
B	1	0				
C	1	1	0			
D	2	1	1	0		
E	2	1	1	1	0	
F	3	2	2	1	1	0

Figure 7

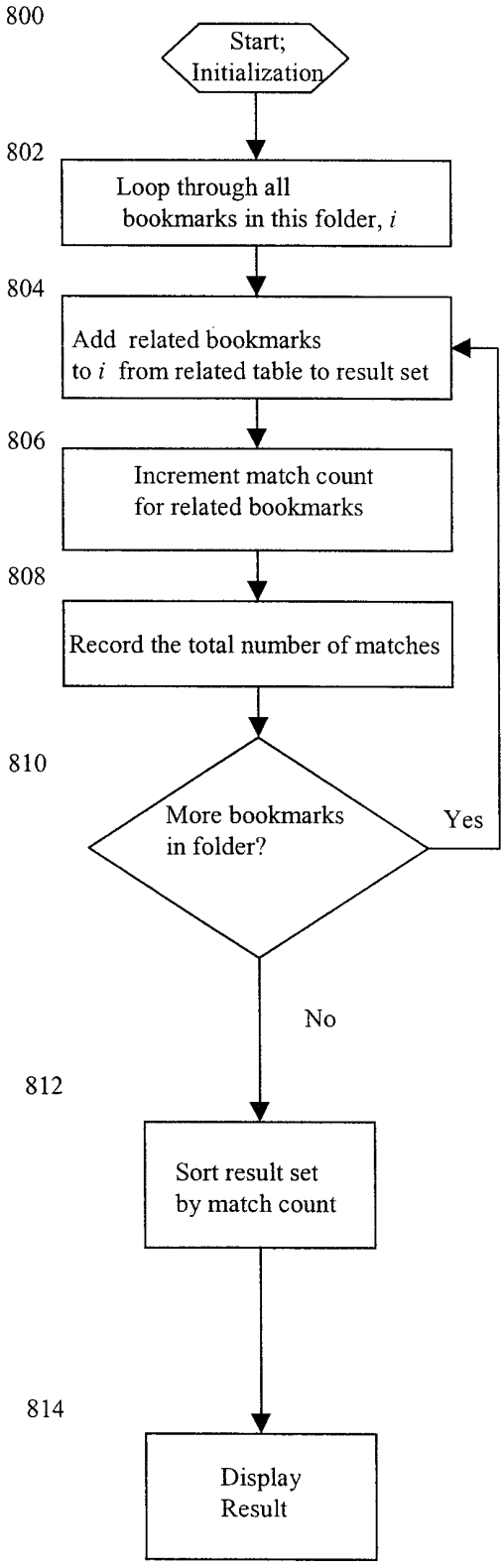

800
Start;
Initialization
802
Loop through all
bookmarks in this folder, i
804
Add related bookmarks
to i from related table to result set
806
Increment match count
for related bookmarks
808
Record the total number of matches
810
More bookmarks
in folder?
Yes
No
812
Sort result set
by match count
814
Display
Result

Figure 8

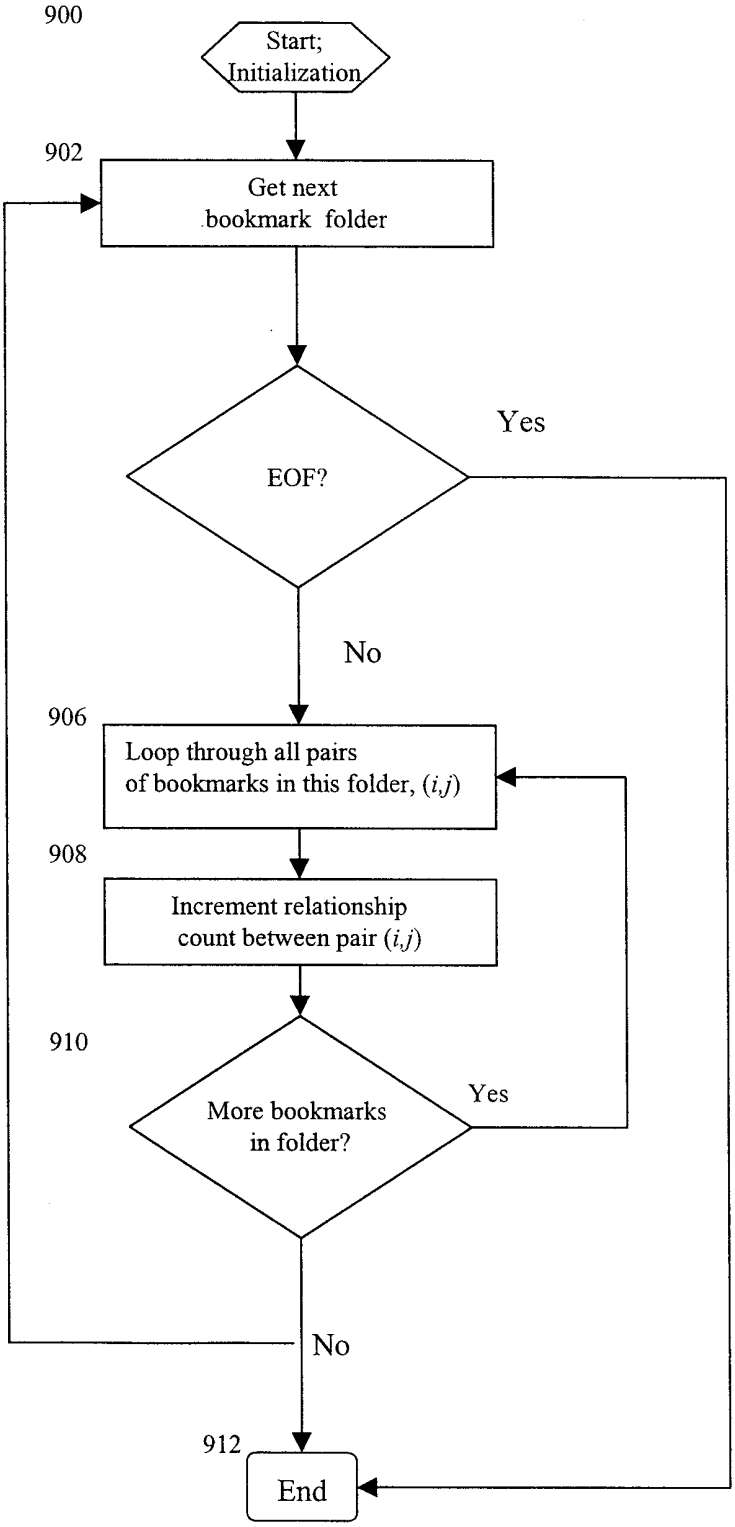

900
Start;
Initialization
902
Get next
bookmark folder
EOF?
Yes
No
906
Loop through all pairs
of bookmarks in this folder, (i,j)
908
Increment relationship
count between pair (i,j)
910
More bookmarks
in folder?
Yes
No
912
End

Figure 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/04588

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : GO6F 13/00, 13/14, 17/00, 17/30, 17/60

US CL : Please See Extra Sheet.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 235/375; 345/329; 705/26, 27; 707/2, 3, 10, 100, 104, 200, 203, 506; 709/1, 201, 217, 218, 219, 235

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A, P	US 5,956,027 A (KRISHNAMURTHY) 21 SEPTEMBER 1999, col. 1-4.	1-41
A, E	US 6,052,687 A (MIURA ET AL.) 18 APRIL 2000, col. 4-12	1-41
A, E	US 6,064,979 A (PERKOWSKI) 16 MAY 2000, col.4-21,	1-41
A, E	US 6,052,717 A (REYNOLDS ET AL.) 18 APRIL 2000, col. 3-15.	1-41
A, P	US 6,023,701 A (MALIK ET AL.) 08 FEBRUARY 2000, col. 2-8	1-41
A, P	US 5,895,461 A (DE LA HUERGA ET AL.) 20 APRIL 1999, col. 4-12.	1-41

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
26 MAY 2000	06 JUL 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer GLEN BURGESS Telephone No. (703) 305-4792

Form PCT/ISA/210 (second sheet) (July 1998) ★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/04588

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

235/375; 345/329; 705/26, 27; 707/2, 3, 10, 100, 104, 200, 203, 506; 709/1, 201, 217, 218, 219, 235