

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
2 August 2001 (02.08.2001)

PCT

(10) International Publication Number
WO 01/55835 A1

(51) International Patent Classification⁷: **G06F 7/00**, 17/30

(74) Agent: **WHITE, Jason, C.**; Brinks Hofer Gilson & Lione,
P.O. Box 10087, Chicago, IL 60610 (US).

(21) International Application Number: PCT/US01/00332

(22) International Filing Date: 5 January 2001 (05.01.2001)

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, 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, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/494,818 31 January 2000 (31.01.2000) US

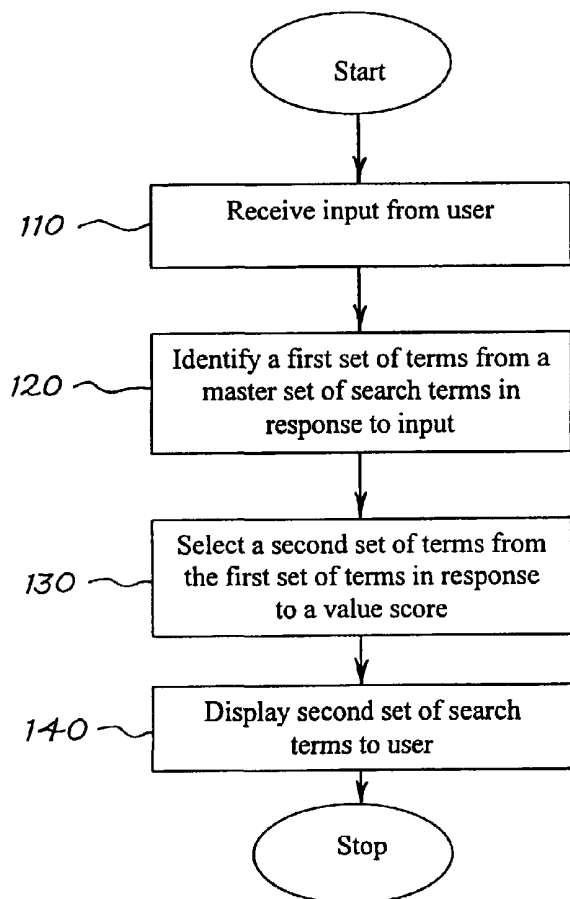
(71) Applicant: **GOTO.COM, INC.** [US/US]; 140 West Union Street, Pasadena, CA 91103 (US).

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, 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, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

(72) Inventor: **KRAVETS, Alexander, N.**; 13600 Marina Pointe Drive, Unit 1203, Marina Del Rey, CA 90292 (US).

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR GENERATING A SET OF SEARCH TERMS



(57) Abstract: A method and system for providing a set of search terms in response to a user input (110) are disclosed. A first set of search terms is selected from a master set of search terms based upon a match between the input and the search terms or based upon a predefined association between the input and the search terms (120). A second set of search terms is selected from the first set of search terms in response to a value score that is established for each of the search terms (130). The value score is selected based at least in part upon the amount of revenue that each search term generates for the system's operator.



WO 01/55835 A1



Published:

— with international search report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND SYSTEM FOR GENERATING A SET OF SEARCH TERMS

FIELD OF THE INVENTION

The present invention relates to a method and system for generating a set of
5 search terms in response to an input provided by a user. More particularly, the
present invention relates to a method and system for generating a set of search terms
in response to an input provided by a user and a value score that is established for
each search term.

BACKGROUND

10 Search engines are commonly used to search the information available on
computer networks such as the World Wide Web to enable users to locate
information of interest that is stored within the network. To use a search engine,
a user typically enters one or more search terms that the search engine uses to
generate a listing of information, such as web pages, that the user is then able to
15 access and utilize. The information resulting from the search is commonly identified
as a result of an association that is established between the information and one or
more of the search terms entered by the user. Different search engines use different
techniques to associate information with search terms and to identify related
information. These search engines also use different techniques to provide the
20 identified information to the user. Accordingly, the likelihood of information being
found as a result of a search varies depending upon the search engine used to
perform the search.

This uncertainty is of particular concern to web page operators that make
information available on the World Wide Web. In this setting, there are often
25 several web page operators that are competing for the same group of potential views
or customers. Accordingly, a web page's ability to be identified as the result of
a search is often important to the success of a web page. Therefore, web page
operators often seek to increase the likelihood that their web page will be seen as the
result of a search.

One type of search engine that provides web page operators with a more predictable method of being seen as the result of a search is a “pay for performance” arrangement where web pages are displayed based at least in part upon a monetary sum that the web page owner has agreed to pay to the search engine operator. The web page operator agrees to pay an amount of money, commonly referred to as the bid amount, in exchange for a particular position in a set of search results that is generated in response to a user’s input of a search term. A higher bid amount will result in a more prominent placement in a set of search results. Thus, a web page operator may attempt to place high bids on one or more search terms to increase the likelihood that their web page will be seen as a result of a search for that term. However, there are many similar search terms, and it is difficult for a web page operator to bid on every potentially relevant search term. Likewise, it is unlikely that a bid will be made on every search term. Accordingly, a search engine operator may not receive any revenue from searches performed using certain search terms for which there are no bids.

In addition, because the number of existing web pages is ever increasing, it is becoming more difficult for a user to find relevant search results. The difficulty of obtaining relevant search results is further increased because of the search engine’s dependency on the search terms entered by the user. The search results that a user receives are directly dependent upon the search terms that the user enters. The entry of one search term may not result in relevant search results, while the entry of only a slightly different search term can result in relevant search results. Accordingly, the selection of search terms is often an important part of the search process. However, current search engines do not enable a search engine operator to provide specific search terms, such as those that will produce relevant search results, to a user. A system that overcomes these deficiencies is needed.

SUMMARY OF THE INVENTION

In accordance with one embodiment of the invention, a set of search terms is provided to a user in response to an input that is provided by the user. In response

to the input, a first set of search terms is generated from a master set of search terms where a portion of each of the search terms in the first set match the input. A second set of search terms is then selected from the first set of search terms based upon a value score that is established for each search term. The value score can be
5 established based, at least in part, upon how much revenue is generated by the search term. The second set of search terms can then be displayed to the user.

According to another embodiment of the invention, in response to input provided by the user, a first set of search terms is generated where the terms are associated with the input, but do not match the input. A second set of search terms is
10 then selected from the first set of search terms based upon a value score that is established for each search term. The value score can be established based, at least in part, upon how much revenue is generated by the search term.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a system for providing a set of search terms to
15 a user of a preferred embodiment.

FIG. 2 flow chart of a method for providing a set of search terms to a user of a preferred embodiment.

FIG. 3 is a more detailed flow chart of a portion of the method depicted in FIG. 2.

20 FIG. 4 is a more detailed flow chart of a portion of the method depicted in FIG. 2.

FIG. 5 is a more detailed flow chart of a portion of the method depicted in FIG. 2.

FIG. 6 depicts an arrangement for displaying a set of search terms of a first
25 preferred embodiment.

FIG. 7 depicts an arrangement for displaying a set of search terms of a second preferred embodiment.

FIG. 8 depicts an arrangement for displaying a set of search terms of a third preferred embodiment.

FIG. 9 is a flow chart of a method for providing a set of search terms to a user of a preferred embodiment.

FIG. 10 depicts an arrangement for activating a system for providing a set of search terms of a preferred embodiment.

5 FIG. 11 depicts an arrangement for deactivating a system for providing a set of search terms of a preferred embodiment.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

Referring now to FIG. 1, a system 10 for generating a set of search terms in response to an input provided by a user is shown. The system 10 comprises a client 20, a website server 30, a predictive search server 40, a database 50, and an event system monitor 70. The system 10 is preferably configured as a client/server architecture. The client 20 preferably comprises a personal computer but can comprise a workstation or any other suitable computer of any size. The client 20 preferably implements a web browser such as Navigator, Explorer, Netscape, or Mosaic, which enables a user to access a “pay for performance” search engine such as the one operated by GoTo.com at www.goto.com or as described in pending U.S. Patent Application No. 09/322,677, which is incorporated herein by reference. The system 10 can be used in conjunction with other types of search engines as well.

The client 20 preferably includes a cache memory 22 that can be used in conjunction with a web browser to store information received from the search server 40, as described below. The cache memory 22 is preferably incorporated with the client 20, but can also be external to the client 20. The client 20 is preferably coupled within the website server 30 via a network 60, such as the Internet. Alternatively, the client 20 can be coupled with the website server 30 via other networks such as a local area network (LAN), a wide area network (WAN), or a regional or other form of network. The phrase “coupled with,” as used herein, means coupled either directly or indirectly via one or more intervening elements.

The client 20 is preferably operative to communicate with the website server 30 using a communication protocol such as HyperText Transfer Protocol (HTTP) or any other suitable protocol. The client 20 is also preferably operative to request a service that is provided by the website server 30 or the predictive search server 40 without having to know any working details about the website server 30 or the predictive search server 40. The client 20 is preferably operative to support JavaScript, which can also be known as ECMAScript, such as the one depicted in Appendix A, which enables the client 20 to generate requests, store information in the cache 22, and display information to a user. The information is preferably displayed through the use of an Internet web browser in an appropriate HTML subdocument element such as an IFRAME element of the HTML4.0 specification or an ILAYER/LAYER element of the Netscape Navigator.

The website server 30 preferably comprises a computer that is coupled with the client 20, the predictive search server 40, and the event system monitor 70. Alternatively, the website server 30 can comprise a workstation or any other suitable computer of any size or a plurality of computers connected in a network configuration. The website server 30 preferably comprises a Website Dynamo Servlet in a standard Dynamo Application Server but can comprise a plurality of Website Dynamo Application Servers. The website server 30 is preferably operative to implement a search engine such as the one operated by GoTo.com at www.goto.com or as described in pending U.S. Patent Application No. 09/322,677. The website server 30 is preferably operative to process requests transmitted from the client 20 and is preferably operative to forward the requests to the predictive search server 40 via an Internet protocol such as the Unreliable Datagram Protocol (UDP).

The predictive search server 40 preferably comprises a computer that is coupled with the website server 30 and the database 50. Alternatively, the predictive search server 40 can comprise a workstation or any other suitable computer of any size or a plurality of computers connected in a network configuration. The predictive search server 40 preferably includes a cache memory 42 that can be used

to store information such as a value score file, as described below. The cache memory 42 is preferably incorporated with the predictive search server 40, but can also be external to the search server 42. The predictive search server 40 is operative to maintain a master set of search terms. As used herein, the phrase “search term”
5 means one or more individual words, numbers, identifiers, terms, or any other form of data. The master set of search terms preferably comprises a set of terms used for use with a search engine for searching the information available on the World Wide Web. The master set of search terms can also comprise other sets of information such as library of congress book numbers, a list of car parts for one or
10 more makes of cars, or a list of the makes and models of cars or boats, for example.

The predictive search server 40 preferably maintains a list of precomputed HTML/JavaScripts that correspond to the search terms in the master set. These scripts can be used to display information to a user as a result of a search. The
15 predictive search server 40 is also operative to receive and store a value score file from the database 50 that lists the value score for each of the search terms within the master set.

The predictive search server 40 is preferably operative to implement a computer program such as the Java program listed in Appendix B attached hereto
20 that enables it to respond to requests from the client 20. The predictive search server 40 is operative to generate UDP protocol responses in response to UDP requests from the client 20. One such UDP request is a search request that can cause the predictive search server 40 to identify one or more sets of search terms from the master set of search terms in response to an input provided by the user, as
25 described herein. A single predictive search server 40 can be used in conjunction with one or a plurality of Website Dynamo Application Servers. A plurality of predictive search servers 40 can also be used.

While the client 20, the website server 30, and the predictive search server 40 are depicted as separate elements, they can be implemented on a single computer. In
30 addition, the functions of the client 20, the website server 30 and the predictive

search server 40 can be combined in a single computer or distributed among many computers.

The database 50 preferably comprises a statistics database that is coupled with the predictive search server 40 and the event system monitor 70. The database 50 preferably comprises a Sun Solaris Server and Oracle database software, but any suitable hardware and software can be used. The database 50 is operative to maintain a list of value scores for the master set of search terms. The database 50 is also operative to receive statistical information relating to the search terms in the master set and update the value scores in response to the statistical information. The database 50 can receive the statistical information from the event system monitor 70. The value scores can be automatically updated in real time, near real time, or at given intervals of time, such as daily, weekly, or monthly. The value scores can be transmitted to the predictive search sever 40 in the form of a value score file.

The value score for each search term can be determined in any number of ways. In a preferred embodiment, a value score comprises a number or index that is between 1 and 100 and is calculated as a composite based upon a number of factors. In a preferred embodiment, where the system 10 is used in conjunction with a pay for performance search engine, the index is a composite in which 85% of the composite is the revenue that that term has generated for the search engine operator and 15% of the composite is the number of time the term has been used in a search in a certain time period. The revenue is preferably calculated by multiplying the bidded amount for a search term and the number of times that search term is entered. The time period for the number of searches preferably comprises one month but can comprise any period of time. The value score can also comprise one or more letters or other symbols.

In alternative embodiments, the value score can be calculated using different percentages of the same factors. In addition to the above factors, the value score can be calculated using one or more of the following factors: the frequency at which a search term is used, the clickthrough rate of a search term,

the paid clickthrough rate of a search term, a demographic profile, and a psychographic profile. The clickthrough rate preferably represents the frequency at which users select or click on search results for a given search term. The frequency can be determined by dividing the total number of search results for a search term that are selected by a user by the total number of searches for a search term that are requested by all the users. This can be used as a measure of how often a user receives relevant information as the result of a search. The demographic profile preferably represents certain characteristics or features of a user that can be used to alter the search terms that are display to that user. The psychographic profile preferably represents certain preferences or tendencies of a user. For example, the psychographic profile can include the user's sensitivity to price of an item such as a compact disc or the user's preference in music (i.e. rap music versus opera).

The event system monitor 70 is preferably coupled with the website server 30 and the database 50. The event system monitor 70 monitors the events occurring on the website server 30 to determine various parameters such as the total number of searches requested by all users of the site, the number of times each search term is entered by a user, the number of times a user selects one of the search results presented to them, the amount of revenue generated by a search term in a pay for performance type search engine, or other suitable parameters. The event system monitor collects the data required to determine various parameters and is able to generate reports detailing the different parameters that can be sent to the database 50.

The system depicted in FIG. 1 can be used to implement the method depicted in FIG. 2. Assume for purposes of this example that a user is utilizing a web browser operating on the client 20 to search information available in a network such as the World Wide Web, for example. Also assume that the user is utilizing a pay for performance search engine such as the one that is provided by GoTo.com at www.goto.com, which requires the user to provide an input to initiate the search. Upon receipt of an input from a user (act 110), a first set of

search terms is identified from a master set of search terms in response to the input (act 120). The input received from the user preferably comprises three alphanumeric characters entered by the user using a keyboard at the client 20, but can consist of any number of characters as well as any other suitable input such as voice input or other input as discussed herein. A second set of search terms is then selected from the first set of search terms in response to a value score for each search terms in the first set (act 130). The second set of search terms can then be displayed to the user (act 140).

Referring now to FIG. 3, the act of receiving an input from a user (act 120, FIG. 2) preferably comprises the following acts. When a user at the client 20 provides an input, a request is automatically generated by the client 20 and is transmitted to the website server 30 (act 210, FIG. 3). The request includes the input provided by the user and requests a response from the predictive search server 40. Upon receipt of the request, the website server 30 forwards the request to the predictive search server 40 (act 220).

Referring now to FIG. 4, the act of identifying a first set of search terms from a master set of search terms in response to the input provided by the user (act 120, FIG. 2) preferably comprises the following acts. After receiving the request from the website sever 30, the predictive search server 40 identifies a first set of search terms where the first three characters of each search term match the three character input provided by the user (act 230, FIG. 4). The first set of search terms preferably comprises 500 search terms but can comprise any number of search terms. The first set of search terms can then be stored in the cache memory 42 that is coupled with the predictive search server 40 (act 240). In alternative embodiments, the predictive search server 40 can respond to various inputs as described herein and can identify a first set of search terms by identifying search terms that have, for example, less than or greater than three matching characters.

In an alternative embodiment, after the first set of search terms has been selected by the search server 40 (act 230, FIG. 4) the first set of search terms can be transmitted to the client 20 and can be stored in the cache memory 22. The

precomputed HTML/JavaScript for each search term can also be transmitted from the predictive search server 40 to the client 20. This process is commonly referred to as aggressive read ahead caching and limits the amount of information that must be transferred from the predictive search server 40 to the client 20 during
5 subsequent stages of the present method. The first set of search terms can then be accessed by either the client 20 or the predictive search server 40 to determine a second set of search terms as described herein.

In an alternative embodiment, the act of identifying a first set of search terms from a master set of search terms in response to the input provided by the
10 user (act 120, FIG. 2), can comprise the following acts. After receiving the request from the website sever 30, the predictive search server 40 identifies a first set of search terms where the first three characters of each search term do not match the three character input provided by the user, but where the search terms are associated with the input provided by the user. The predictive search server 40
15 preferably identifies this first set of search terms through a reverse indexing process the associates search terms with possible inputs that a user can provide. This allows for the identification of search terms that would likely be of interest to a user but that do not share the same characters as the search terms.

For example, if a user provided the input "ope" presumably seeking
20 information about an opera, the search term "New York Opera" could be provided. In other examples, the input of a portion of a recording artist's name could result in the display of songs performed by the artist; the input of a portion of an author's name could result in the display of books by that author; the input of a portion of the name of a state could result in the display of cities located within that state; the
25 input of a portion of a class name could result in the display of students in that class; the input of a portion of a school name could result in the display of teachers in that school; the input of a portion of a symptom of an illness could result in the display of drugs associated with that symptom; the input of a portion of a type of animal could result in the display of animals in that classification; and the input of
30 a portion of the name of an article of manufacture, the name of the manufacturer,

or the serial number of the article could result in the display of manufacturers of that type of item. Other relationships are also possible.

In a further alternative embodiment, the act of identifying a first set of search terms from a master set of search terms in response to the input provided by the user (act 120, FIG. 2), can comprise the following acts. The predictive search server 40 can identify a first set of search terms where one or more of the search terms is identified through either a match between the first three characters of the search term and the input provided by the user and one or more of the search terms is identified through a reverse indexing process.

Referring now to FIG. 5, the act of selecting a second set of search terms from the first set of search terms in response to a value score for each search term (act 130, FIG. 2) preferably comprises the following acts. The predictive search server 40 analyzes the value scores that correspond to the search terms collected in the first set (act 250). This can be done through the use of the term value file that can be generated by the database 50 and transmitted to the predictive search server 40, as described herein. The predictive search server 40 can then select a number of search terms, such as six, from the first set of search terms that have the highest value scores (act 260). In alternative embodiments, greater than or less than six search terms can be selected. As described herein, the value score can be determined in various ways.

In an alternative embodiment, the act of displaying the second set of search terms to the client 20 (act 140, FIG. 2) preferably comprises transmitting a precomputed HTML/JavaScript for each term in second set of search terms from the predictive search server 40 to the client 20. In some embodiments this step is not necessary as the precomputed HTML/JavaScript for each of the terms in the first set of search terms is stored at client 20 through the aggressive read ahead caching procedure. The second set of search terms can be displayed to the user in nearly real time while the user is providing the input but before the user finishes providing the input. For example, the second set of search terms can be displayed to the user before the user finishes typing all of the characters of a search term.

Referring now to FIGS. 6-8, the second set of search terms can be displayed to the user as shown. The second set of search terms can be displayed as a list 300 in conjunction with a search engine input screen as shown in FIG. 6.

The second set of search terms can also be displayed in a single list 310 in conjunction with a search engine results page as shown in FIG. 7. Alternatively, the second set of search terms can be displayed in a first list 310 and a second list 320 in conjunction with a search engine results page as shown in FIG. 8. The second set of search terms can also be displayed in other variations and combinations.

The second set of search terms are preferably displayed in alphabetical order but can be displayed in descending order based on the value score for each search term, ascending order of the value scores, other orders based upon the value score, or in virtually any other order irrespective of the value score. The search terms preferably appear as links as shown in FIG. 6 that, when selected by the user, will result in the automatic display of the search results associated with the search term. Alternatively, the second set of search terms can be displayed as a list of terms that when selected by the user appear in the search query box of a search engine. Alternatively, the second set of search terms can be displayed as a drop-down list that can include links to search results.

The method of FIG. 2 can be repeated if the user provides additional input. For example, if the user provides a further input such as a fourth character, a revised set of search terms can be identified from the first set of search terms, as described herein, and can be displayed to the user, as described herein.

Alternatively, in response to the receipt of additional input provided by a user (act 410, FIG. 9), a revised first set of search terms can be identified from the master set of search terms (act 420), as described herein. A revised second set of search terms can then be identified from the revised first set of search terms (act 430), as described herein and can be displayed to the user (act 440), as described herein.

Referring now to FIGS. 10 and 11, the method and system described herein can be used in conjunction with a search engine such as the one provided by

GoTo.com at www.goto.com and can be activated and deactivated by a user through the selection or deselection of one or more links 330 as shown in FIGS. 10 and 11. The links 330 can be displayed in conjunction with a traditional search engine entry and can be selected by a user by “clicking” on one of the links 330 to
5 turn the system on or off.

The foregoing method and system enable a set of search terms to be provided to a user in a predictive or suggestive fashion. When the user provides a partial input, the method and system can present the user with a set of search terms that match the partial input or that are somehow related to the partial input. The
10 set of search terms can be displayed to the user is nearly real time while the user is providing the input and before user finishes providing the input. The user can then select from the search terms predicted or suggested to them without having to provide the remainder of his intended input. This enables a search engine operator to provide a user with search terms that are likely to produce the best search results
15 as well as the search terms that will generate the most revenue for the search engine operator.

It is to be understood that a wide range of changes and modifications to the embodiments described above will be apparent to those skilled in the art and are contemplated. It is therefore intended that the foregoing detailed description be
20 regarded as illustrative rather than limiting, and that it be understood that it is the following claims, including all equivalents, that are intended to define the spirit and scope of the invention.

Appendix A

```

!-- BEGIN PREDICTIVE SEARCH HTML FRAGMENT -->

SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.1">!--
//This has to precede the IFRAME/ILAYER
function defaultSB(){
    return '<html><body bgcolor=#ffcc00><center><img src=/images/home/ms.gif><center></body></html>'
}
/---</SCRIPT>

<IFRAME ID=SB SRC="javascript:parent.defaultSB()" onFocus="passFocus()"
    HEIGHT=88 SCROLLING=NO MARGINHEIGHT=0 MARGINWIDTH=0 HSPACE=0 VSPACE=0 FRAMEBORDER=0>
    <ILAYER NAME=OSB><LAYER ID=SB HEIGHT=88 SRC="javascript:parent.defaultSB()"></LAYER></ILAYER>
</IFRAME>

<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript1.1">!--
//Initialize some useful globals
var cache = new Array() //store a hashtable of prefixes and their cache entries
var values = new Array() //store a hashtable of prefixes and their cache entry values
var noCache = new Array() //store a hashtable of negative responses to prefixes
var tabFromUp = false

//Get a portable ref to the IFRAME or ILAYER
SB = (document.all) ? document.all.SB : document.OSB.document.SB;
//Get a portable ref to Keywords search box
KW = document.search.Keywords

function passFocus(){
    if(SB.document.links.length < 1)
        return
    if(SB.document.links[0]){
        if(tabFromUp) {
            SB.document.links[0].focus()
            tabFromUp = false
        }
        else {
            SB.document.links[SB.document.links.length-1].focus()
        }
    }
}

function change(newValue){
    newValue = canonicalize(newValue)
    //alert('in change with newValue = ' + newValue + ' length= ' + newValue.length)
    if(newValue.length < 3) {
        showDefault()
    }else if(newValue.length == 3) {
        if(cache[newValue] != null) {
            scrollCache(newValue)
        }
        else {
            refreshAndScrollCache(newValue)
        }
    }else {
        //alert('in change with newValue = ' + newValue + ' length= ' + newValue.length)
        scrollCache(newValue)
    }
}
//change

```



```

function refreshAndScrollCache(prefix) { //force reload
    SB.document.location.replace('/d/sc/?prefix=' + escape(prefix) + '&source=generic')
}

function makeHref(term) {
    return (
        '<a href="/d/search/?Keywords=' + escape(term) +
        '&type=suggested" target=_parent>' + term + '</a><br>\n')
}

function indexOfMinimum(array) {
    var minimum = array[0]
    var minIndex = 0
    for (var i in array) {
        if(array[i] < minimum) {
            minimum = array[i]
            minIndex = i
        }
    }
    return minIndex
}

function showDefault() {
    if(SB.document.images == null || SB.document.images.length == 0)
        SB.location.replace('javascript:parent.defaultSB()')
}

function canonicalize(value) {
    while(value.charAt(0) == ' ' || value.charAt(0) == '+')
        value = value.substring(1)
    return value.toLowerCase()
}

function setCache(key, c, v) {
    //alert('in set cache with key = ' + unescape(escape(key)) + ' KW.value = ' + KW.value)
    if(c == null) {
        showDefault() //dont cache cache timeouts
    } else if(c.length == 0) {
        noCache[key] = 'none' //do cache negative responses
        showDefault()
    } else {
        for(var i in c) c[i] = unescape(c[i]) //clean up and save away this prefix cache
        cache[key] = c
        values[key] = v
        scrollCache(canonicalize(KW.value)) //scroll to the latest cursor
    }
}

function scrollCache(prefix) {
    //util arrays
    var key = prefix.substring(0,3)
    //alert('in scrollCache with prefix = ' + prefix + ' key = ' + key)
    if(cache[key] == 'none') return //if no cache then return early
    var list = new Array()
    var listValues = new Array()

    for (var i in cache[key]) {
        //Skip prefix and those that don't start with prefix
        if(prefix >= cache[key][i] || cache[key][i].indexOf(prefix, 0) != 0)
            continue
        if(list.length < 5) { //prepopulate list with first candidates
            list[list.length] = cache[key][i]
            listValues[listValues.length] = values[key][i]
        } else {
            var mi = indexOfMinimum(listValues)
            if(values[key][i] > listValues[mi]) {

```

```
        list[mi] = cache[key][i]
        listValues[mi] = values[key][i]
    }
    } //else
} //for all in cache[key]
//At this point list should contain 5 highest value words > prefix,
//just alpha sort and show it
list.sort()
if(list.length > 0){
    //give it a style
    SB.document.write(
        '<html><head><style> A {font-size:9pt; font-family: Helvetica, Arial}' +
        'A:hover {color:red} A:active {color:red}</style></head><body bgcolor=#ffcc00>\n')
    for(var k in list)
        SB.document.write(makeHref(list[k]))
    SB.document.write('</body></html>')
    SB.document.close()
} else
    showDefault()
}
//--></script>
```

Appendix B

```

package com.go2.search.predictive;

import java.util.*;
import java.net.*;
import java.io.*;

/**
 * This class implements the in-memory Predictive Search cache
 * At boot time a file with cache entries and their corresponding values
 * is loaded and processed by this Class and thereafter a service
 * is made available on a UDP port to quickly serve client requests
 * @author Alex Kravets <alex@goto.com>
 */
public class PredictiveServer {

    /**
     * Bind a UDP port and listen for requests
     * @param args java.lang.String[] - Argument: /cache/file/path udpPortToBind
     */
    public static void main(String args[]) {
        //Simple usage check
        if(args.length != 3) throw new IllegalArgumentException(
            "Usage: java com.go2.search.predictive.PredictiveServer /path/to/cache perPrefixMax udpPortToBind");

        System.out.println("Starting Predictive Server ... \nTotal JVM Mem Usage Before cache load: " +
            (Runtime.getRuntime().totalMemory()/1000000 + " Megabytes: ");

        try {
            //Load our cache from a file and maintain some stats about it
            long start = System.currentTimeMillis();
            loadCache(args[0]);
            pruneCache(Integer.parseInt(args[1]));
            buildMap();
            long end = System.currentTimeMillis();
            System.gc(); //Free up some RAM

            printStats(start, end);
            serviceLoop(Integer.parseInt(args[2]));
        } catch (Exception e) {
            System.out.println("Failed with: " + e);
            e.printStackTrace();
        }
    }

    private static void loadCache(String cacheFile) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader(cacheFile), 64*1024);
        while(true){
            String line = reader.readLine();
            if(line == null) break; //Indicates we're done with this file
            int valueSep = line.lastIndexOf(VSEP);
            //encode for safe transport in HTML/JavaScript
            String phrase = line.substring(0, valueSep).trim();
            //Ignore phrases that are less than MIN_PHRASE_LENGTH chars
            //greater than MAX_PHRASE_LENGTH or contain '\\'
            int len = phrase.length();
            if(len < MIN_PHRASE_LENGTH || len > MAX_PHRASE_LENGTH || phrase.indexOf('\\') != -1)
                continue;
            phraseLength += phrase.length();
            phraseCount++;
            String prefix = phrase.substring(0, PREFIX_SIZE);
        }
    }
}

```

```

//encode for safe transport in HTML/JavaScript
phrase = cleanPhrase(phrase);
String value = line.substring(valueSep + 1).trim();
ArrayList prefixLines;
//Add a clean line into subBuffers for processing
if((prefixLines = (ArrayList)prefixMap.get(prefix)) != null) {
    prefixLines.add(phrase + VSEP + value); //will need to disassemble again
}else{ //if we haven't seen this prefix before than create a holder for it
    prefixLines = new ArrayList();
    prefixMap.put(prefix, prefixLines);
}
}
} //while more lines to read
reader.close();
} //loadCache

private static void pruneCache(int keep) {
    //For every prefix in the cache, leave highest "keep" in values
    //and then resort alphabetically, remember and drop those prefixes
    //that had less than MIN_PREFIX_ENTRIES values in them
    ArrayList small = new ArrayList();
    //System.out.println("Prefixes with number of entries exceeding " + keep + ": ");
    Iterator prefixIterator = prefixMap.keySet().iterator();
    while(prefixIterator.hasNext()){
        String prefix = (String)prefixIterator.next();
        ArrayList segment = (ArrayList)prefixMap.get(prefix);
        int size = segment.size();
        prefixListSize += size;
        //Sort entries in the descending order of values if necessary
        if(size > keep) { //if prefix too big
            //System.out.println("'" + prefix + "' by " + (segment.size() - keep) + ". ");
            Collections.sort(segment, new Comparator(){
                public int compare(Object a, Object b) {
                    String aLine = (String)a;
                    String bLine = (String)b;
                    int aValue = Integer.parseInt(aLine.substring(aLine.lastIndexOf(VSEP) + 1));
                    int bValue = Integer.parseInt(bLine.substring(bLine.lastIndexOf(VSEP) + 1));
                    return bValue - aValue; //induce descending ordering
                }
                public boolean equals(Object another) { return super.equals(another); }
            });
            //Drop those of lower value
            segment.subList(keep, segment.size()).clear();
        }
        //Put in desc alpha order for further consumption by buildMap
        Collections.sort(segment);
        //Remember prefixes that are too short for subsequent removal from prefixMap
        if(size < MIN_PREFIX_ENTRIES)
            small.add(prefix);
    } //for all prefixes
    //Actually drop all those that are too short
    System.out.println("Number of prefixes with fewer than "+MIN_PREFIX_ENTRIES+" entries: "+small.size());
    for(int i = 0; i < small.size(); i++) {
        //System.out.println("Dropping " + small.get(i));
        prefixMap.remove(small.get(i));
    }
} //pruneCache

private static void buildMap() {
    //Precompute HTML/Javascript for every prefix in the map
    Iterator prefixIterator = prefixMap.keySet().iterator();
    while(prefixIterator.hasNext()){
        prefixCount++;
        String prefix = (String)prefixIterator.next();
        //get segment and build Html for it
        ArrayList segment = (ArrayList)prefixMap.get(prefix);
        String html = HTML_HEADER;
        //Make sure to add the prefix into it first !!

```

```

StringBuffer entries = new StringBuffer("var c = new Array(" + ESEP + prefix + ESEP);
StringBuffer values = new StringBuffer("var v = new Array(0");
ListIterator lineIterator = segment.listIterator();
while(lineIterator.hasNext()){
    String line = (String)lineIterator.next();
    int valueSep = line.lastIndexOf(VSEP);
    //add javascript String quotes
    entries.append(", " + ESEP + line.substring(0, valueSep) + ESEP);
    values.append(',' + line.substring(valueSep + 1));
} //all in prefix segment
entries.append(");\n");
values.append(");\n");
html += entries;
html += values;
//html += "for(var i in c) c[i] = unescape(c[i]);\n";
html += "parent.setCache(" + ESEP + prefix + ESEP + ", c, v);\n";
html += HTML_FOOTER;
prefixMap.put(prefix, html.getBytes());
} //for all entries
//buildMap

private static void serviceLoop(int port) throws SocketException {
    //Bind our UDP port and set UDP in/out buffers high to improve net I/O
    DatagramSocket socket = new DatagramSocket(port);
    socket.setSendBufferSize(64*1024);
    socket.setReceiveBufferSize(64*1024);
    //Enter the eternal loop
    while(true) {
        try {
            DatagramPacket in = new DatagramPacket(inBuffer, inBuffer.length);
            socket.receive(in); //block here until a request arrives
            byte[] output = lookup(new String(inBuffer, 0, in.getLength()));
            socket.send(new DatagramPacket(output, output.length, in.getAddress(), in.getPort()));
            //System.out.println("Recieved: " + packet.getData().length + " bytes");
        } catch(IOException ioe) { } //Indicates read failure, ignore and try again
    } //eternal while
} //serviceLoop

private static String cleanPhrase(String dirty){
    return URLEncoder.encode(dirty).replace('+', ' ');
}
/*
int apoIndex = -1;
StringBuffer sb = new StringBuffer(dirty);
while((apoIndex = dirty.indexOf(charToEscape, apoIndex + 1)) != -1)
    sb.insert(apoIndex, '\\');
return sb.toString();
*/
//cleanPhrase

private static byte[] lookup(String key) {
    Object data = prefixMap.get(key);
    return (data != null) ? (byte[]) data : NOPREFIX_HTML;
} //lookup

private static void printStats(long start, long end) {
    System.out.print("Total number of entries: " + phraseCount + " ");
    System.out.println("Average cache entry is: " + phraseLength/phraseCount + " characters");
    System.out.print("Total number of good prefixes: " + prefixCount + " ");
    System.out.println("Average size of a good prefix list: " + (prefixListSize/prefixCount) + " entries");
    System.out.print("Time to load cache: " + (end - start)/1000.0 + " seconds.");
    System.out.println(" Rate: " + phraseCount/((end - start + 1)/1000.0) + " per second");
    System.out.println("Total JVM Mem Usage After Cache Load: " +
        (Runtime.getRuntime().totalMemory())/1000000 + " Megabytes: " +

```

```
Runtime.getRuntime().totalMemory()/phraseCount + " bytes per phrase");
//System.out.println("Sample HTML: \n" + new String((byte[])prefixMap.get("afr")));
//printStats

//Constants
private static final int PREFIX_SIZE = 3;
private static final int MIN_PREFIX_ENTRIES = 3;
private static final int MIN_PHRASE_LENGTH = 4;
private static final int MAX_PHRASE_LENGTH = 30;
private static final String HTML_HEADER = "<html><head><script>\n";
private static final String HTML_FOOTER = "</script></head></html>";
private static final char ESEP = ' ';
private static final char VSEP = '\n';
private static final byte[]
NOPREFIX_HTML=(HTML_HEADER+"parent.setCache(\"\", new Array(), new
rray());\n"+HTML_FOOTER).getBytes();

//maps possible prefixes to byte-encoded HTML/JavaScript for them
private static final TreeMap prefixMap = new TreeMap();
//preallocate a single receive buffer for all inbound requests
private static final byte[] inBuffer = new byte[16];

//statistics related stuff
private static int phraseCount;
private static int phraseLength;
private static int prefixCount;
private static int prefixListSize;
//class PredictiveServer
```

I claim:

1. A method of generating a set of search terms in response to an input from a user, the method comprising:

- (a) maintaining a master set of search terms;
- 5 (b) determining a value score for each search term in the master set;
- (c) receiving an input from the user;
- (d) selecting a first set of search terms from the master set of search terms in response to the input received from the user; and
- 10 (e) selecting a second set of search terms from the first set of search terms in response to the value score associated with first set of search terms.

2. The method of claim 1, further comprising arranging the second set of search terms in a search term list.

3. The method of claim 1, further comprising displaying the second set of search terms.

15 4. The method of claim 1, further comprising:

- (f) arranging the second set of search terms in a search term list;
- (g) displaying the search term list with the terms arranged in descending order based upon the value score for each search term.

20 5. The method of claim 1, wherein the value score is determined in response to an amount of revenue generated by each search term.

6. The method of claim 1, wherein the value score for each search term is determined in response to a number of times that the search term is used during a period of time.

25 7. The method of claim 1, wherein the value score for each search term is determined in response to a frequency at which the search term is used.

8. The method of claim 1, wherein the value score for each search term is determined in response to an amount of revenue generated by each search term and a number of times that the search term is used during a period of time.

9. The method of claim 1, wherein the input from the user comprises a plurality of characters.

10. The method of claim 1, wherein (b) comprises receiving statistical information and determining for at least one of the search terms in the master set and
5 determining the value score for the at least one search term in response to the statistical information.

11. The method of claim 1, wherein (d) comprises selecting search terms that contain or match input.

12. The method of claim 1, wherein (d) comprises selecting search terms that
10 are associated with input.

13. The method of claim 1, wherein (e) comprises selecting search terms that have the highest value score.

14. A system for generating a set of search terms in response to an input from a user, the system comprising:

15 a database storing a master set of search terms and a value score for each search;
a server coupled with the database, the serving being operative to select a first set of search terms from the master set of search terms in response to an input received from the user and being operative to select a second set of search terms from the first set of search terms in response to the value score associated with first set of search terms.

20 15. The system of claim 14, further comprising a client coupled with the server.

16. The system of claim 14, further comprising an event system monitor coupled with the server and the database.

17. The system of claim 14, wherein the server comprises a plurality of servers.

25 18. A method of generating a set of search terms in response to an input from a user, the method comprising:

(a) maintaining a master set of search terms;

- (b) determining a value score for each search term in the master set wherein the value score is determined by an amount of revenue generated by each search term;
- (c) receiving an input from the user;
- 5 (d) selecting a first set of search terms from the master set of search terms in response to the input received from the user wherein a portion of each term in the first set of search terms matches the input; and
- (e) selecting a second set of search terms from the first set of search terms in response to the value score associated with first set of search terms.

10 19. The method of claim 18, further comprising displaying the second set of search terms.

20. The method of claim 18, wherein the value score for each search term is determined in response to the amount of revenue generated by each search term and a number of times that the search term is used during a period of time.

15 21. A method of generating a set of search terms in response to an input from a user, the method comprising:

- (a) maintaining a master set of search terms;
- (b) determining a value score for each search term in the master set wherein the value score is determined by an amount of revenue generated by each
- 20 search term;
- (c) receiving an input from the user;
- (d) selecting a first set of search terms from the master set of search terms in response to the input received from the user wherein each term in the first set of search terms is associated with the input; and
- 25 (e) selecting a second set of search terms from the first set of search terms in response to the value score associated with first set of search terms.

22. The method of claim 21, further comprising displaying the second set of search terms.

23. The method of claim 21, wherein the value score for each search term is determined in response to the amount of revenue generated by each search term and a number of times that the search term is used during a period of time.

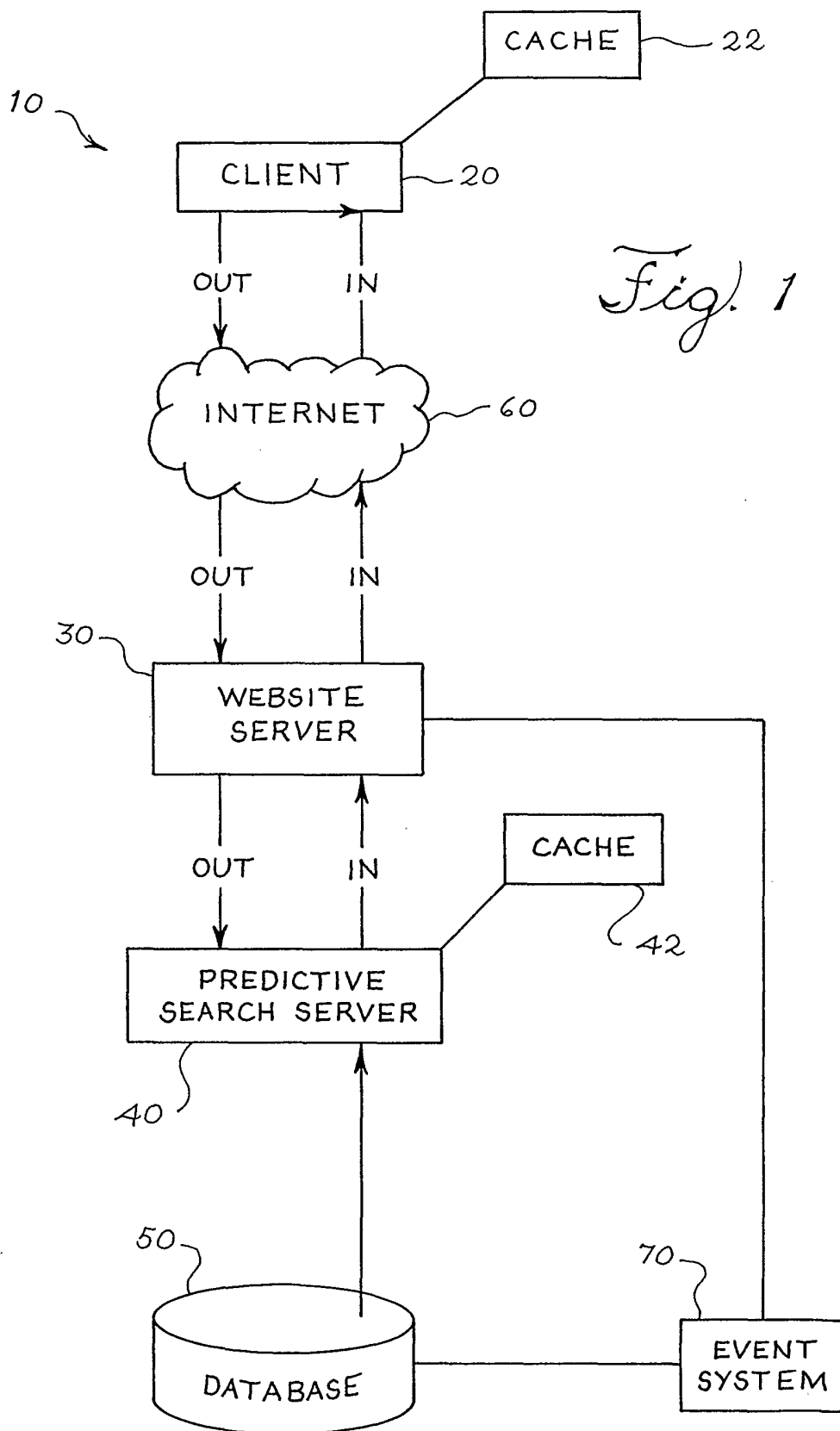


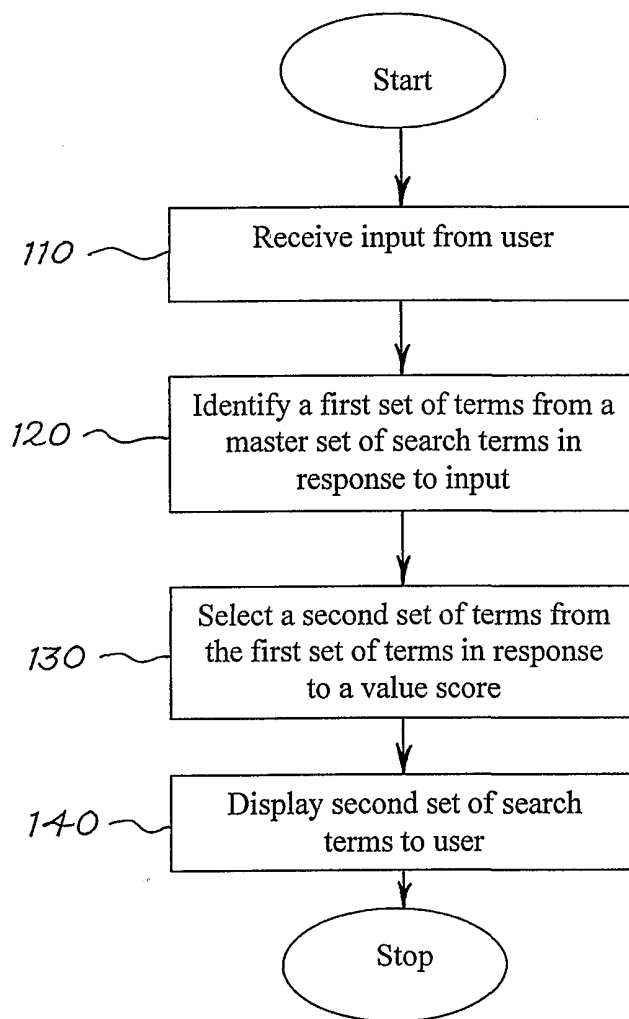
Fig. 2

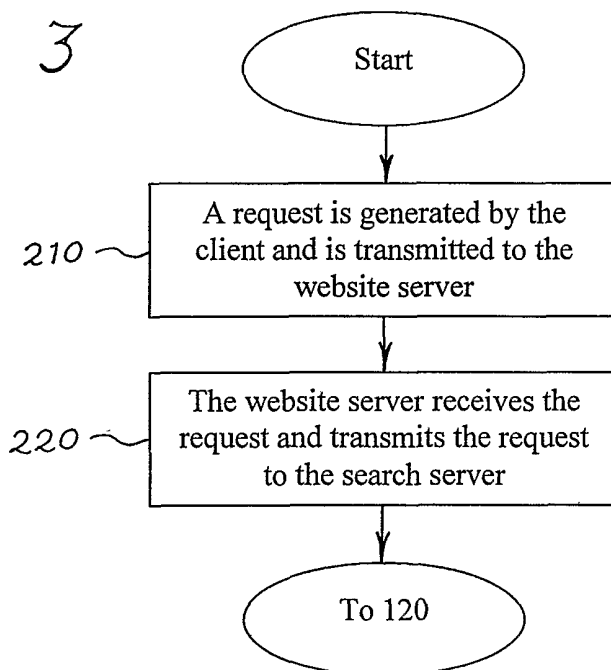
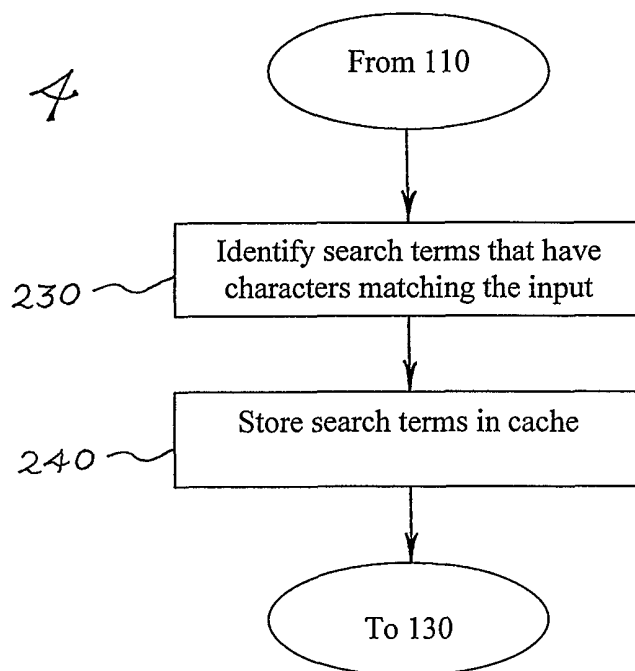
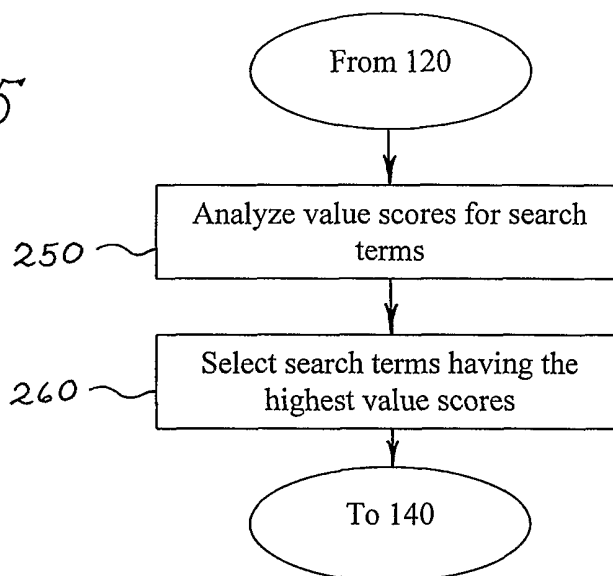
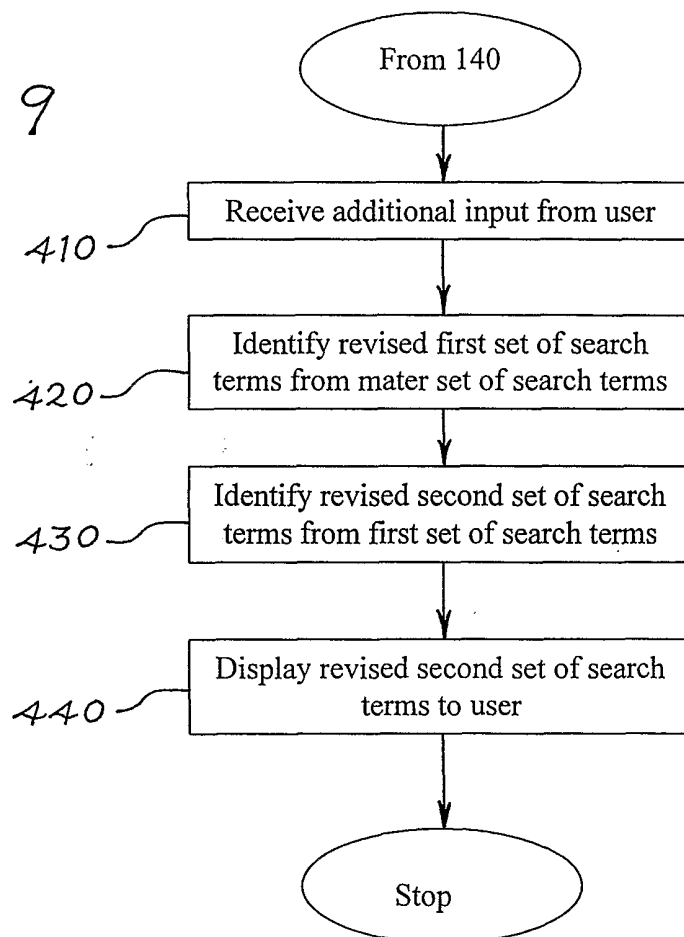
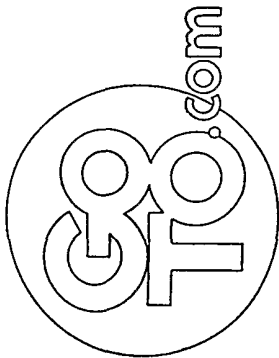
Fig. 3*Fig. 4*

Fig. 5*Fig. 9*



Simply type what you're looking for and GoTo it!

Toile

Find It!

Internet Explorer 5 customized for GoTo users

DOWNLOAD NOW

toile du quebec
toilet
toilet cam
toilet slave
toilet training

SmartSearch : **On / Off**

Computing
Computer Games, Computer Hardware, Software, Web Hosting ...

Finance
Debt, Home Finance, Insurance, Investing, Small Business ...

Reference
Environment, History, Lookup, News, US Govt Agencies ...

Education & Career
K-12, College Education, Financing, Careers, Job Search ...

Health
Diet & Fitness, Drugs & Supplies, Men's Health, Women's Health ...

Shopping
Auto, Books, Computers, Electronics, Sporting Goods ...

Entertainment
Family Attractions, Gambling, Movies & TV, Music, Sports ...

Home Life
Cooking, Gardening, Home Improvement, Pets, Wedding ...

Travel
Adventure Travel, Air Travel, Cruises, Lodging, US Travel ...

List Your Site

Manage Your Account

Make \$\$\$ with GoTo

Investor Relations

About GoTo

300

Fig. 6

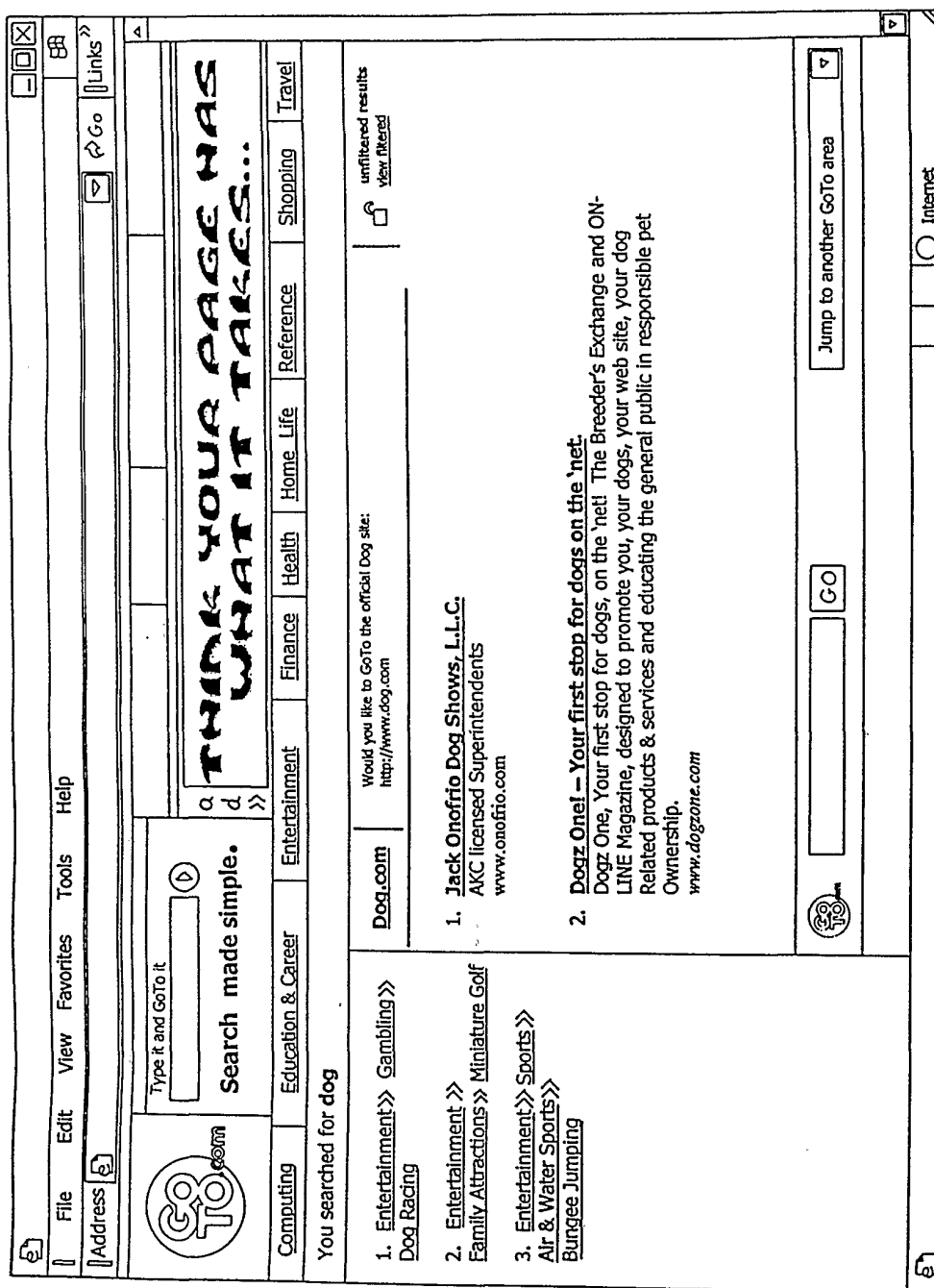


Fig. 7

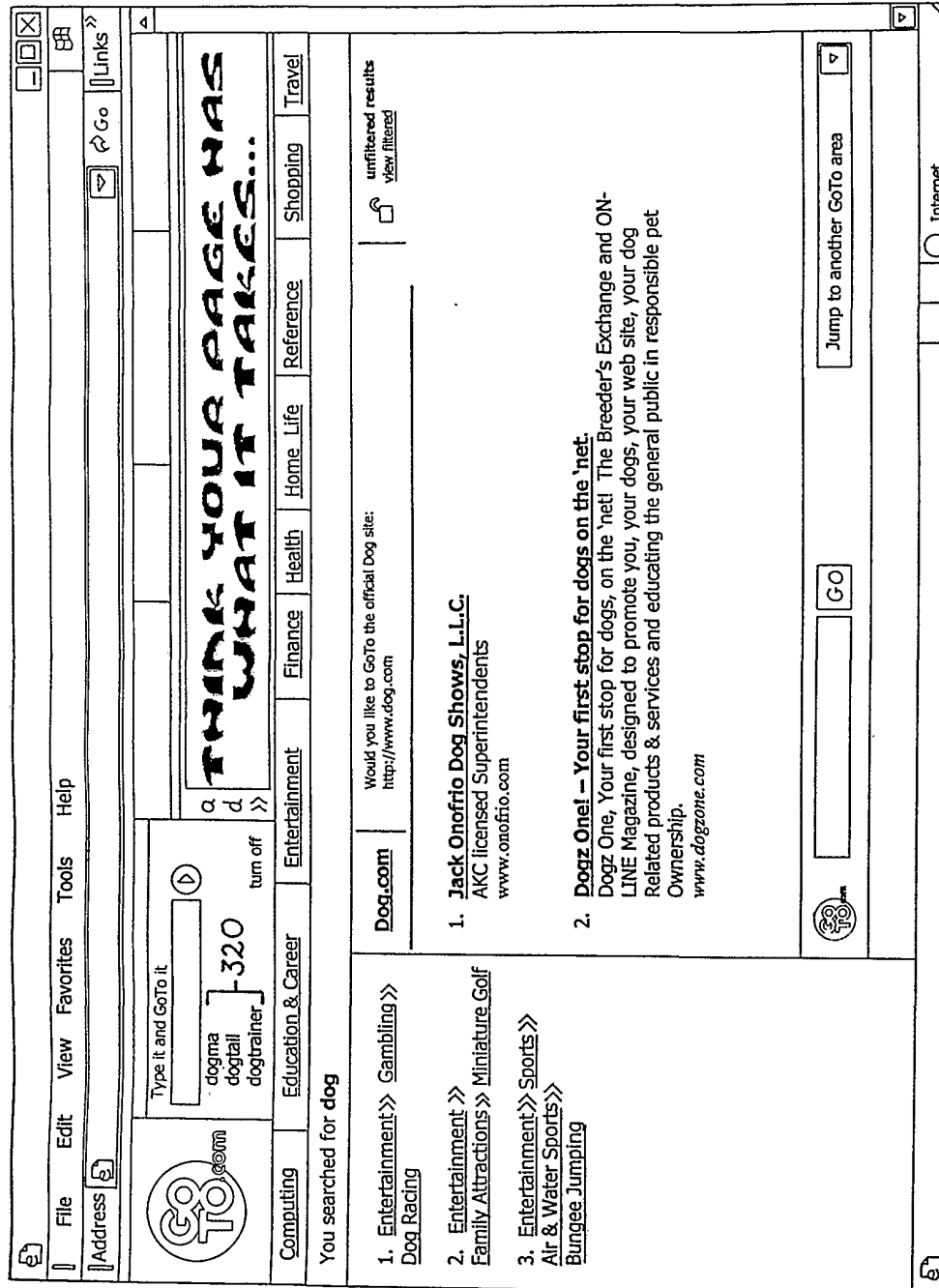


Fig. 8

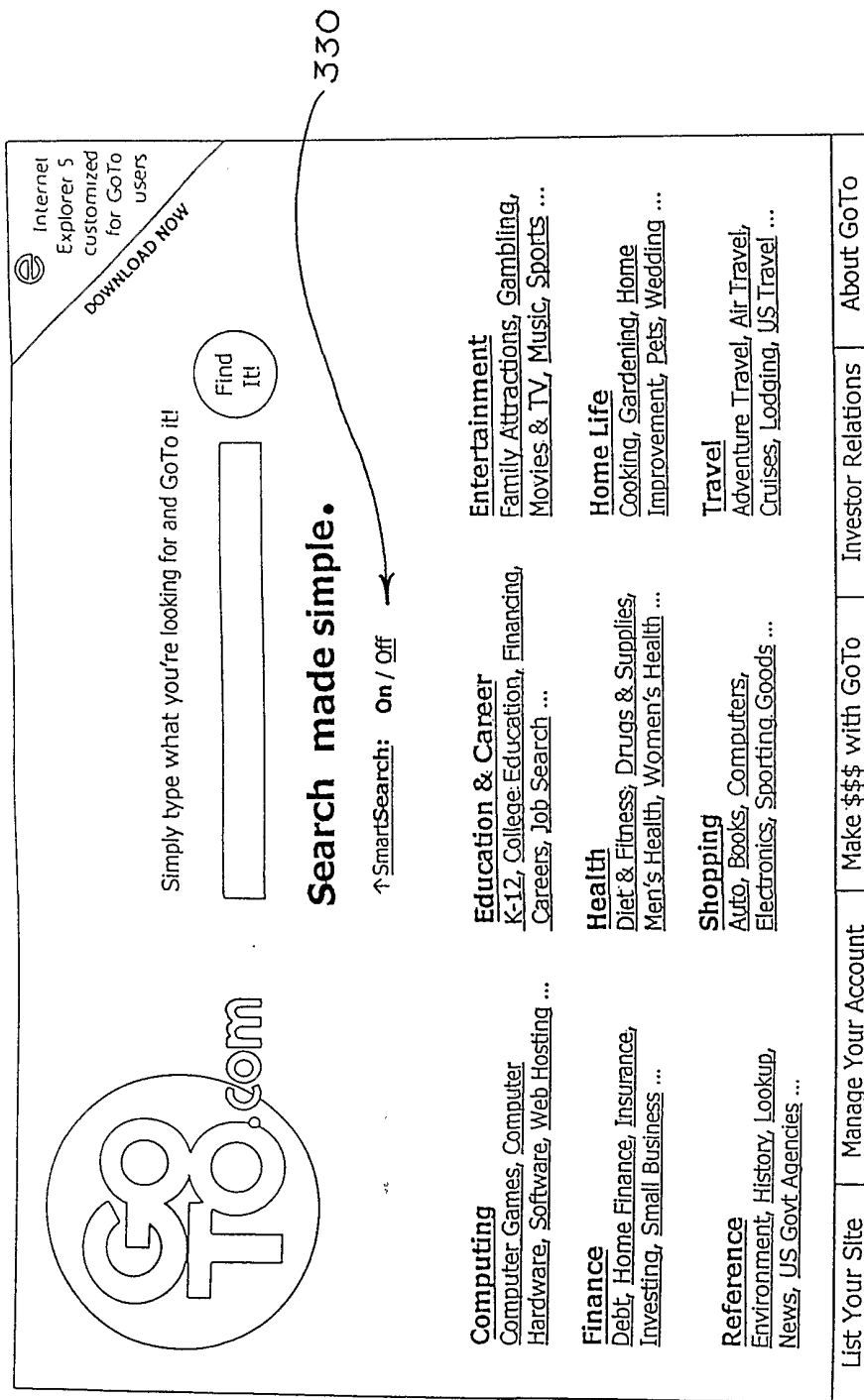


Fig. 10

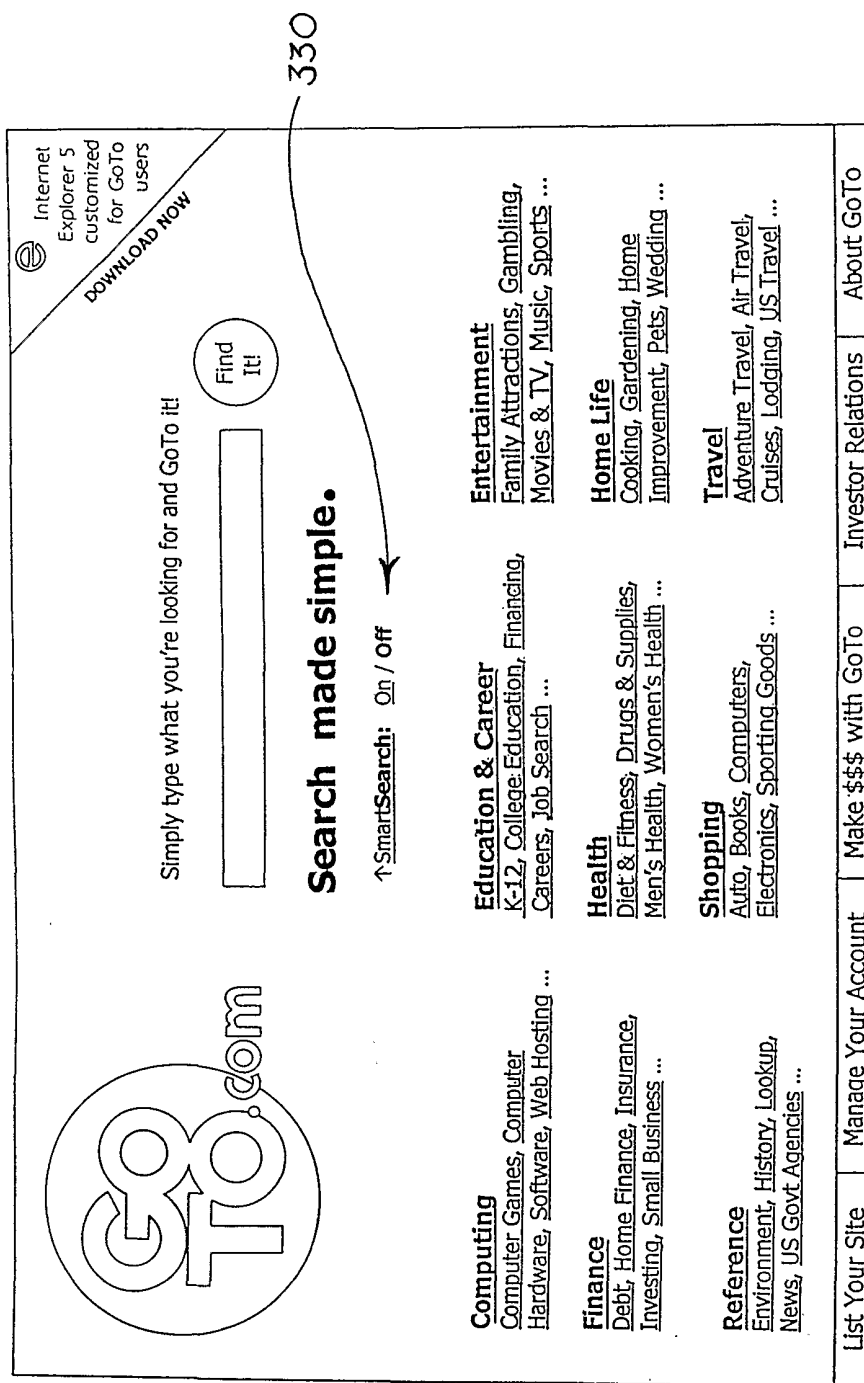


Fig. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/00332

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 7/00, 17/30

US CL :707/3, 4, 5

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. : 707/3, 4, 5

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
NONE

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,206,949 A (COCHRAN et al.) 27 April 1993, the entire paper is relevant	1-23
Y	US 5,715,444 A (DANISH et al.) 03 February 1998, the entire paper is relevant	1-23
Y	US 5,742,816 A (BARR et al.) 21 April 1998, the entire paper is relevant	1-23
Y	US 5,745,894 A (BURROWS et al.) 28 April 1998, the entire paper is relevant	1-23
Y	US 5,819,259 A (DUKE-MORAN et al.) 06 October 1998, the entire paper is relevant	1-23

☒ 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

08 FEBRUARY 2001

Date of mailing of the international search report

09 APR 2001

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

THOMAS BLACK

Telephone No. (703) 305-9707

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/00332

⌚ (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,995,979 A (COCHRAN et al.) 30 November 1999, the entire paper is relevant	1-23
Y	US 6,012,053 A (PANT et al.) 04 January 2000, the entire paper is relevant	1-23