US 20070038603A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2007/0038603 A1**

Guha (43) **Pub. Date: Feb. 15, 2007**

(54) **SHARING CONTEXT DATA ACROSS PROGRAMMABLE SEARCH ENGINES**

(76) Inventor: **Ramanathan V. Guha**, Los Altos, CA (US)

Correspondence Address:
GOOGLE / FENWICK
SILICON VALLEY CENTER
801 CALIFORNIA ST.
MOUNTAIN VIEW, CA 94041 (US)

(21) Appl. No.: **11/202,410**

(22) Filed: **Aug. 10, 2005**

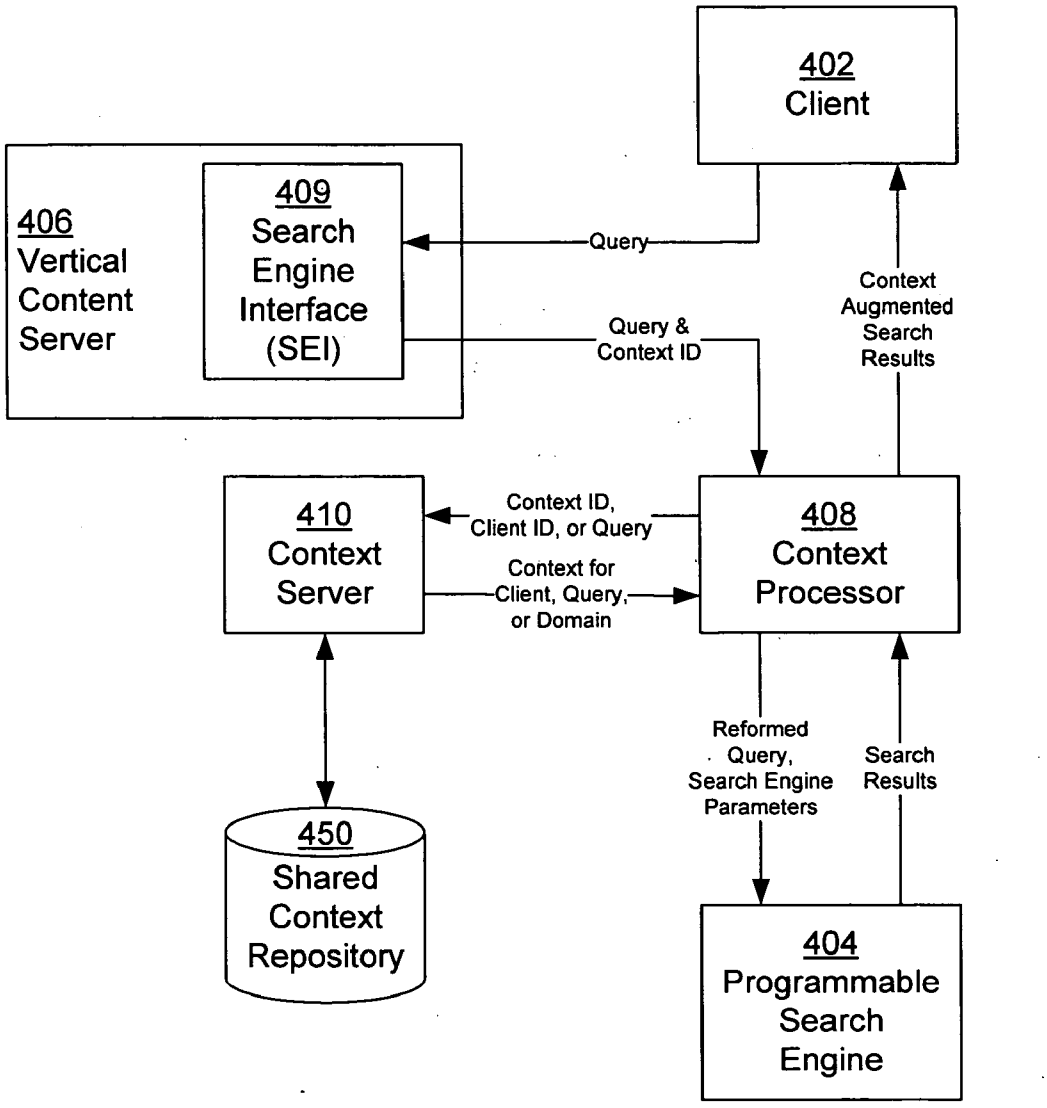
Publication Classification

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** **707/3**

(57) **ABSTRACT**

Context information, obtained by one programmable search engine and relating to a user, is stored in a central shared repository so that it can be used by other programmable search engines, in order to provide more effective targeted search results for the user in the future. Information obtained from other sources, including from the user him- or herself, can also be stored in the repository for future use. A programmable search engine consults the shared repository when processing queries and/or results, so as to incorporate previously stored context information concerning the user.



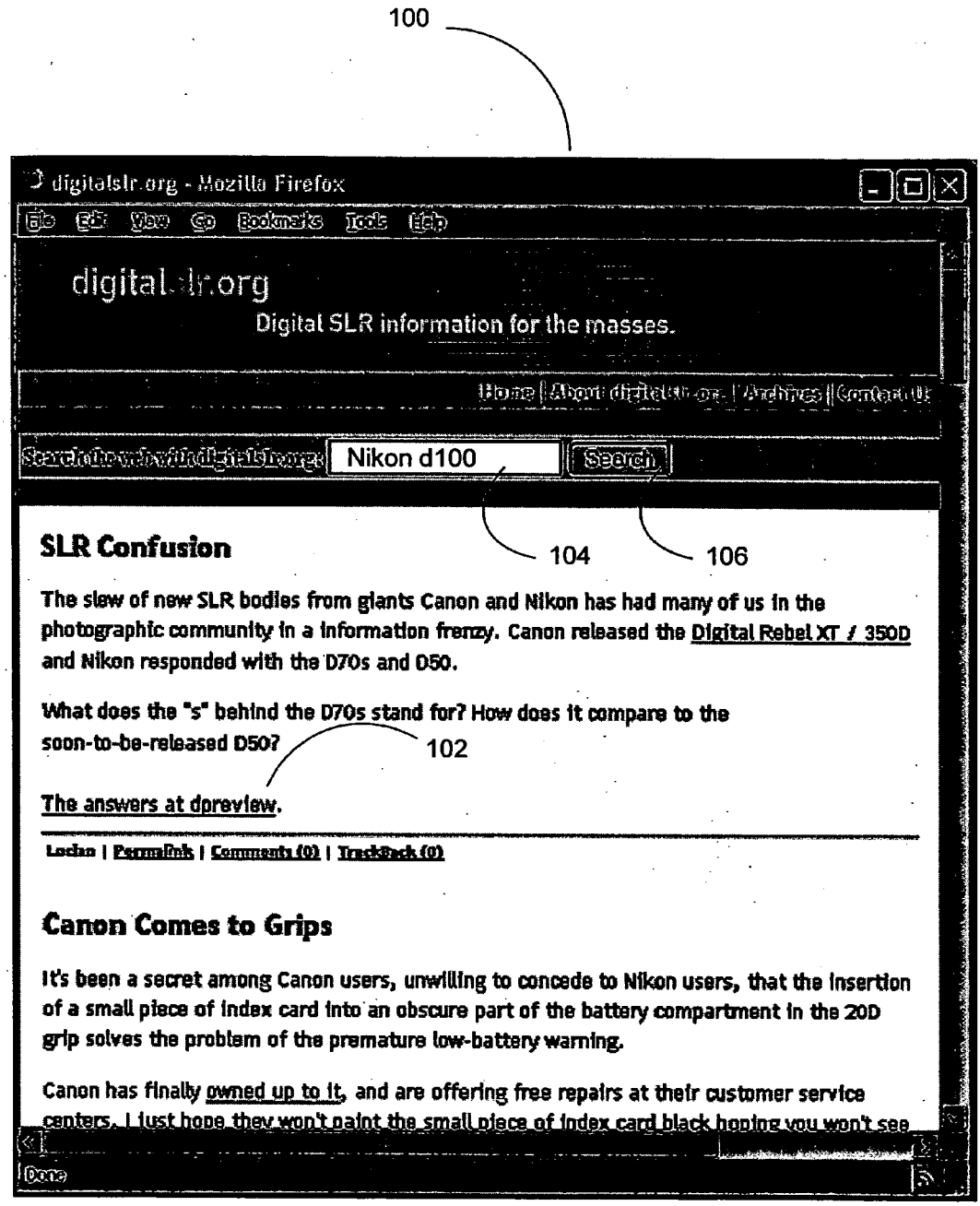

100
digitalslr.org - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
digital-slr.org
Digital SLR information for the masses.
Home | About digital-slr.org | Archives | Contact Us
Search the web with digitalslr.org: Nikon d100 Search
104
106
SLR Confusion
The slew of new SLR bodies from giants Canon and Nikon has had many of us in the photographic community in a information frenzy. Canon released the Digital Rebel XT / 350D and Nikon responded with the D70s and D50.
What does the "s" behind the D70s stand for? How does it compare to the soon-to-be-released D50?
102
The answers at dpreview.
Leehan | Permalink | Comments (0) | TrackBack (0)
Canon Comes to Grips
It's been a secret among Canon users, unwilling to concede to Nikon users, that the insertion of a small piece of index card into an obscure part of the battery compartment in the 20D grip solves the problem of the premature low-battery warning.
Canon has finally owned up to it, and are offering free repairs at their customer service centers. I just hope they won't paint the small piece of index card black hoping you won't see
Done

FIG. 1

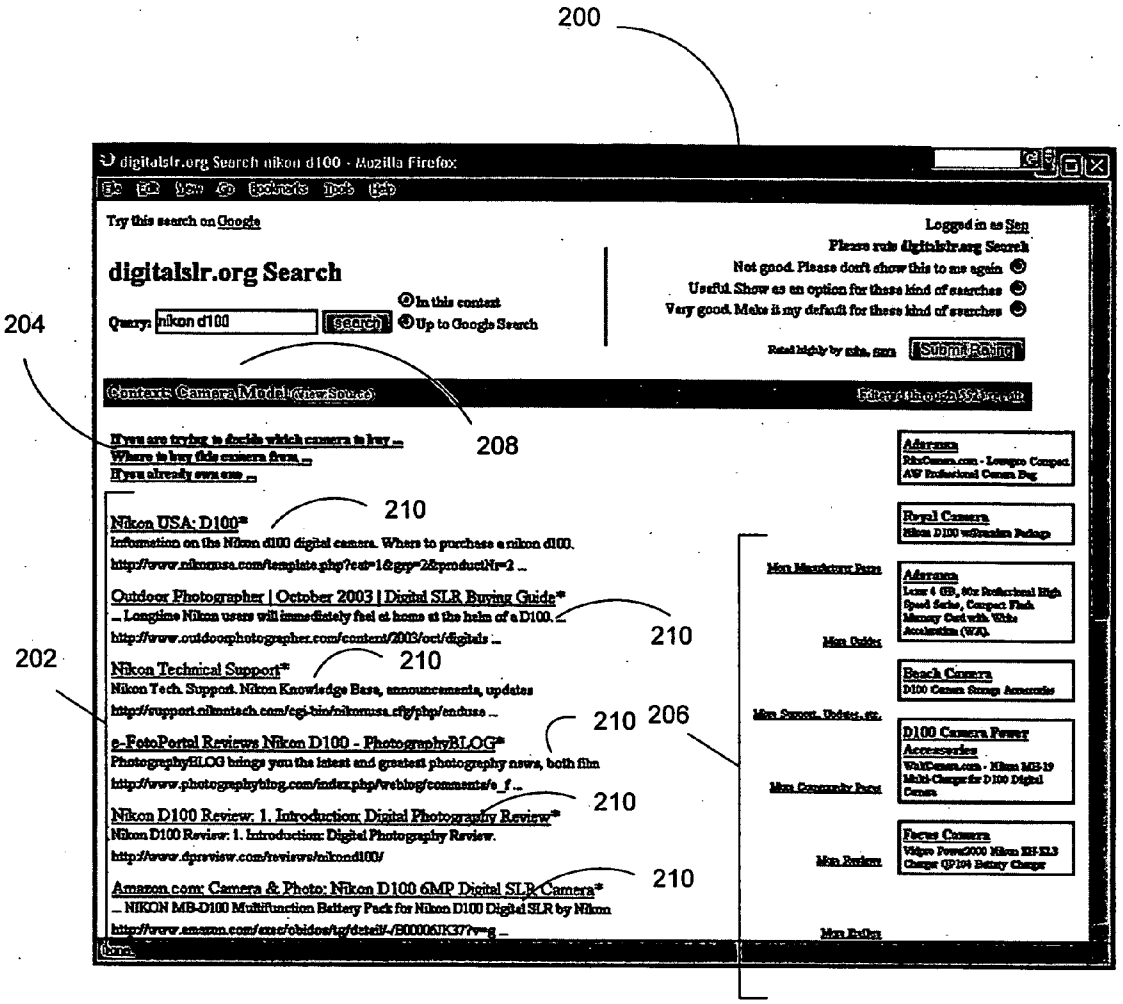

200
digitalslr.org Search nikon d100 - Mozilla Firefox
Try this search on Google
digitalslr.org Search
Query: nikon d100
Search
In this context
Up to Google Search
Logged in as Sen
Please rate digitalslr.org Search
Not good. Please don't show this to me again
Useful. Show as an option for these kind of searches
Very good. Make it my default for these kind of searches
Submit Rating
Contents Camera Model (View Source)
If you are trying to decide which camera to buy ...
Where to buy this camera from ...
If you already own one ...
208
204
210
Nikon USA: D100*
Information on the Nikon d100 digital camera. Where to purchase a nikon d100.
http://www.nikonusa.com/template.php?cat=1&grp=2&product%=2 ...
Outdoor Photographer | October 2003 | Digital SLR Buying Guide*
... Longtime Nikon users will immediately feel at home at the helm of a D100. ...
http://www.outdoorphotographer.com/content/2003/oct/digitals ...
210
202
Nikon Technical Support*
Nikon Tech. Support. Nikon Knowledge Base, announcements, updates
http://support.nikontech.com/cgi-bin/nikonusa.cfg/php/enduse ...
210
206
e-FotoPortal Reviews Nikon D100 - PhotographyBLOG*
PhotographyBLOG brings you the latest and greatest photography news, both film
http://www.photographyblog.com/index.php/weblog/comments/e_f ...
210
Nikon D100 Review: 1. Introduction: Digital Photography Review*
Nikon D100 Review: 1. Introduction: Digital Photography Review.
http://www.dpreview.com/reviews/nikond100/
210
Amazon.com: Camera & Photo: Nikon D100 6MP Digital SLR Camera*
... NIKON MB-D100 Multifunction Battery Pack for Nikon D100 Digital SLR by Nikon
Adorama
Royal Camera
Beach Camera
D100 Camera Power Accessories
Focus Camera
More Manufacturer Pages
More Guides
More Support, Updates, etc.
More Community Posts
More Reviews
More Profiles
Done

FIG. 2

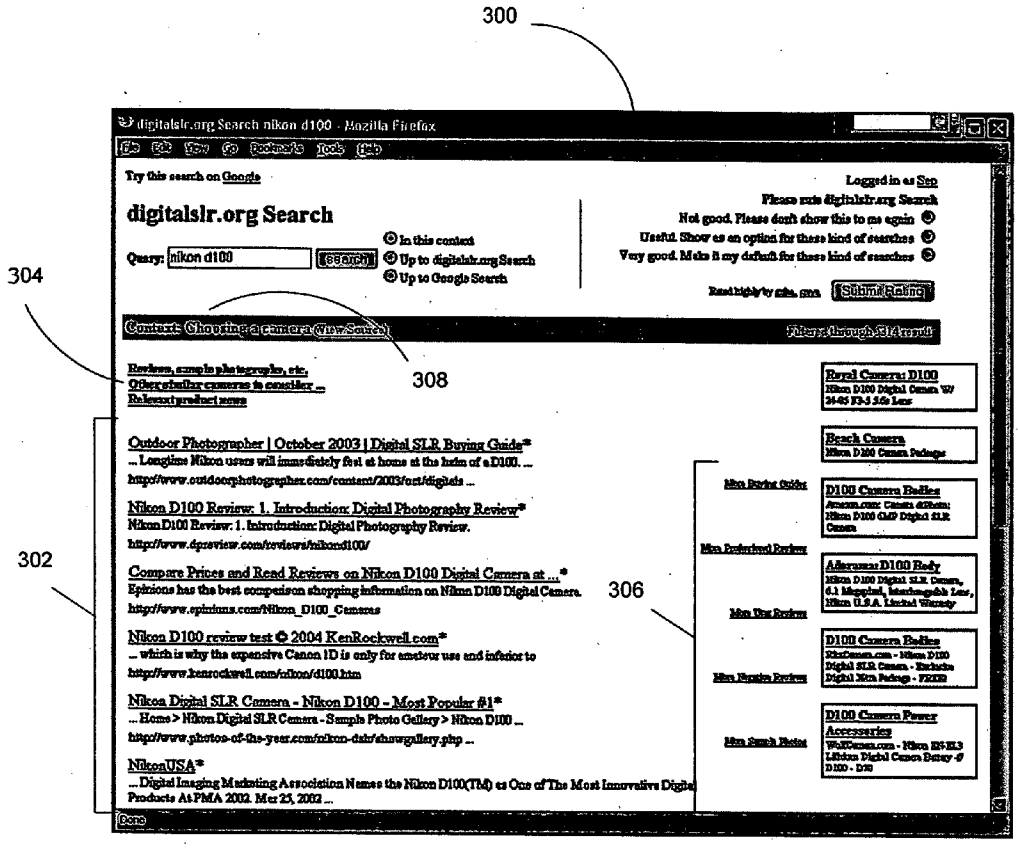

300
304
308
302
306
digitalslr.org Search nikon d100 - Mozilla Firefox
Try this search on Google
digitalslr.org Search
Query: nikon d100
Search
In this context
Up to digitalslr.org Search
Up to Google Search
Logged in as Sep
Please rate digitalslr.org Search
Not good. Please don't show this to me again
Useful. Show as an option for these kind of searches
Very good. Make it my default for these kind of searches
Submit Rating
Context: Choosing a camera (View Source)
Reviews, sample photographs, etc.
Other similar cameras to consider ...
Relevant product news
Outdoor Photographer | October 2003 | Digital SLR Buying Guide*
... Longtime Nikon users will immediately feel at home at the helm of a D100. ...
http://www.outdoorphotographer.com/content/2003/oct/digitals ...
Nikon D100 Review: 1. Introduction: Digital Photography Review*
Nikon D100 Review: 1. Introduction: Digital Photography Review.
http://www.dpreview.com/reviews/nikond100/
Compare Prices and Read Reviews on Nikon D100 Digital Camera at ...*
Epinions has the best comparison shopping information on Nikon D100 Digital Camera.
http://www.epinions.com/Nikon_D100_Cameras
Nikon D100 review test © 2004 KenRockwell.com*
... which is why the expensive Canon 1D is only for amateur use and inferior to
http://www.kenrockwell.com/nikon/d100.htm
Nikon Digital SLR Camera - Nikon D100 - Most Popular #1*
... Home > Nikon Digital SLR Camera - Sample Photo Gallery > Nikon D100 ...
http://www.photos-of-the-year.com/nikon-dslr/showgallery.php ...
NikonUSA*
... Digital Imaging Marketing Association Names the Nikon D100(TM) as One of The Most Innovative Digital Products At PMA 2002. Mar 25, 2002 ...
Royal Camera: D100
Beach Camera
D100 Camera Bodies
Adorama: D100 Body
D100 Camera Bodies
D100 Camera Power Accessories
Done

FIG. 3

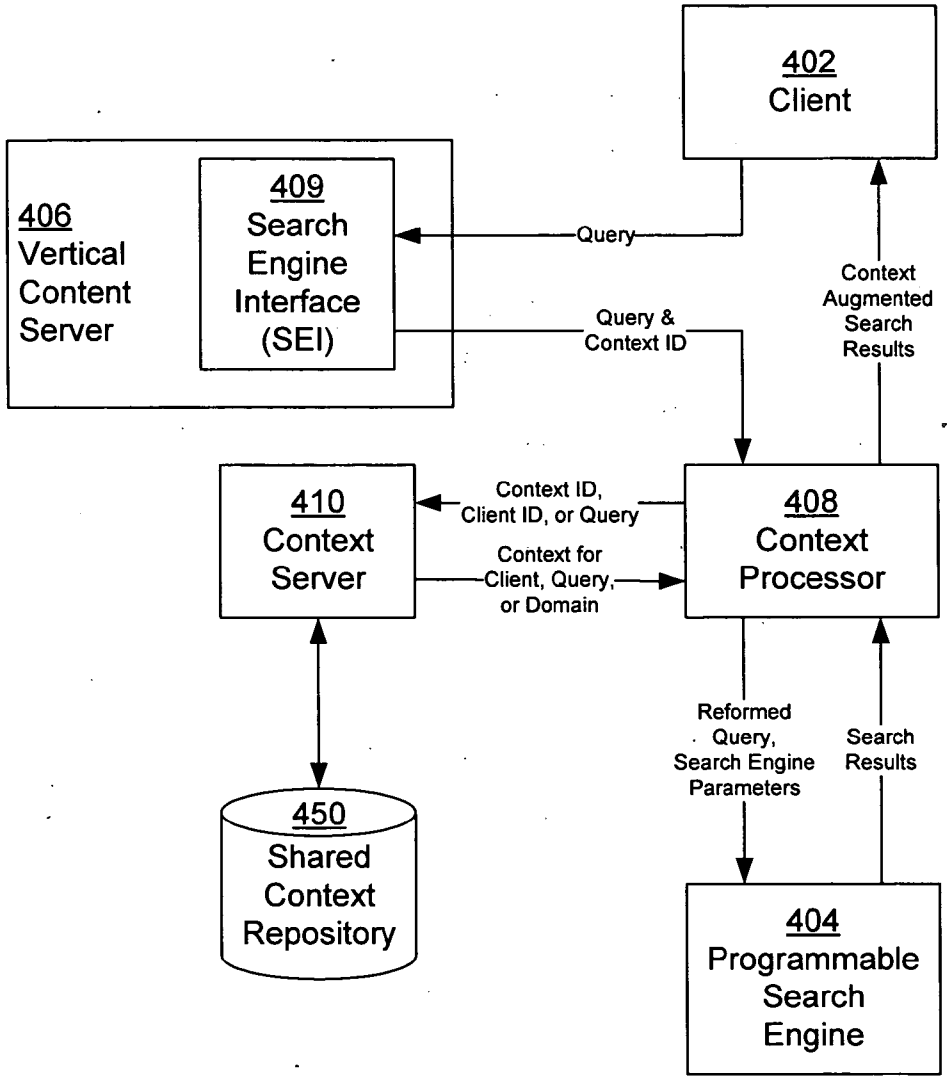

402
Client
406
Vertical
Content
Server
409
Search
Engine
Interface
(SEI)
Query
Query &
Context ID
Context
Augmented
Search
Results
410
Context
Server
Context ID,
Client ID, or Query
Context for
Client, Query,
or Domain
408
Context
Processor
Reformed
Query,
Search Engine
Parameters
Search
Results
450
Shared
Context
Repository
404
Programmable
Search
Engine

FIG. 4

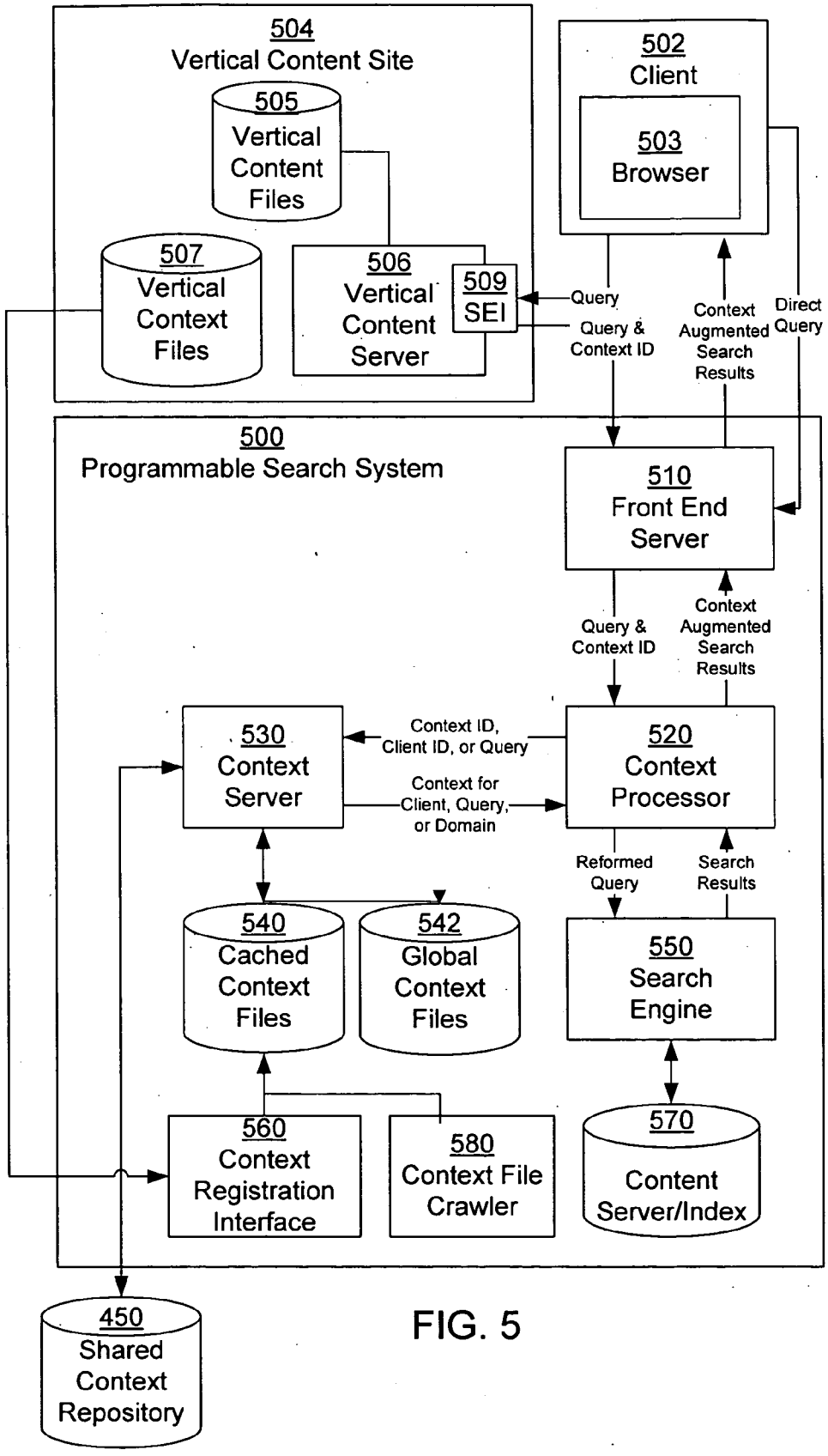

504
Vertical Content Site
505
Vertical Content Files
507
Vertical Context Files
506
Vertical Content Server
509
SEI
502
Client
503
Browser
Query
Query & Context ID
Context Augmented Search Results
Direct Query
500
Programmable Search System
510
Front End Server
Query & Context ID
Context Augmented Search Results
530
Context Server
Context ID, Client ID, or Query
Context for Client, Query, or Domain
520
Context Processor
Reformed Query
Search Results
540
Cached Context Files
542
Global Context Files
550
Search Engine
560
Context Registration Interface
580
Context File Crawler
570
Content Server/Index
450
Shared Context Repository

FIG. 5

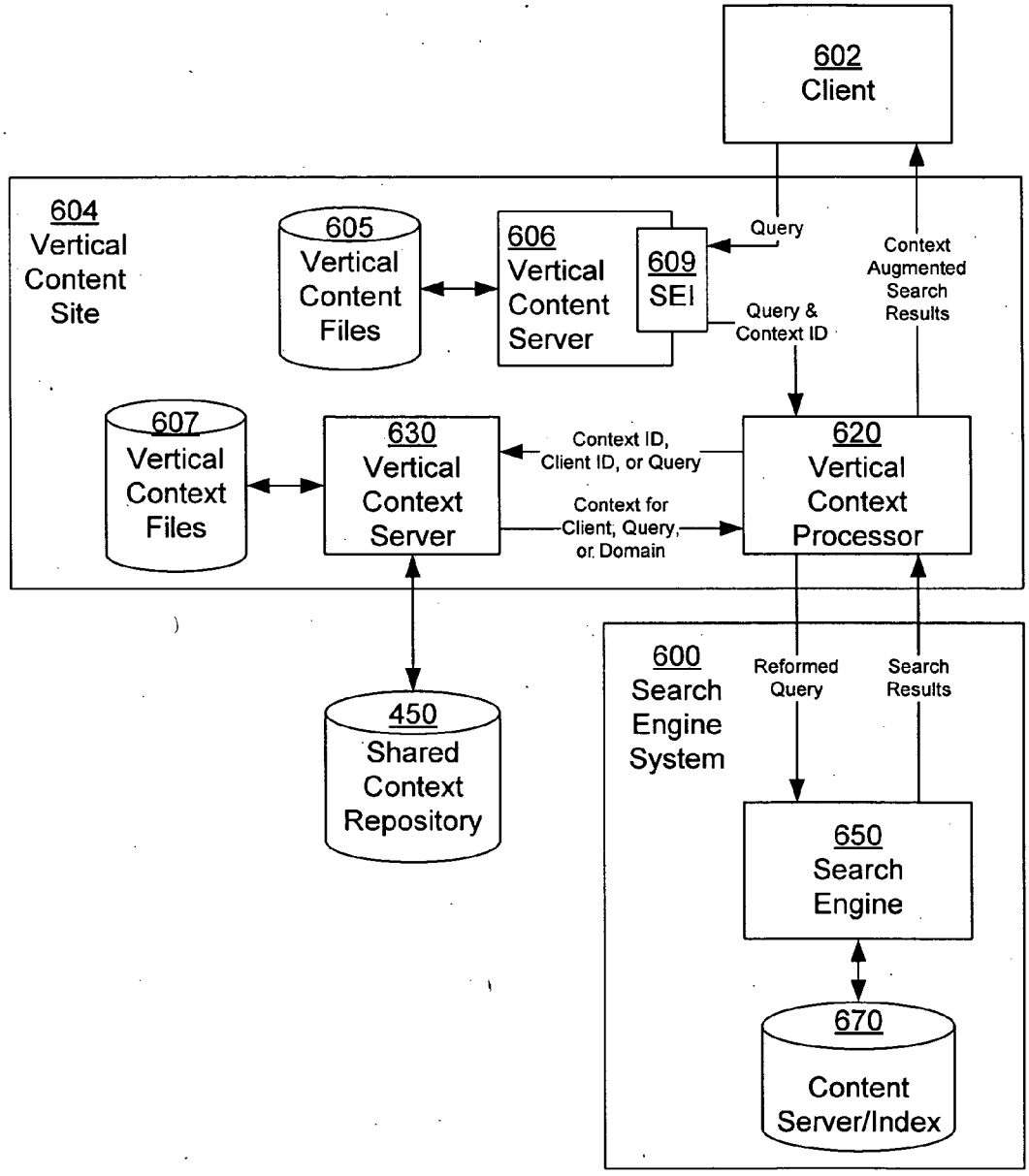

602
Client
604
Vertical
Content
Site
605
Vertical
Content
Files
606
Vertical
Content
Server
609
SEI
Query
Query &
Context ID
Context
Augmented
Search
Results
607
Vertical
Context
Files
630
Vertical
Context
Server
Context ID,
Client ID, or Query
Context for
Client, Query,
or Domain
620
Vertical
Context
Processor
600
Search
Engine
System
Reformed
Query
Search
Results
450
Shared
Context
Repository
650
Search
Engine
670
Content
Server/Index

FIG. 6

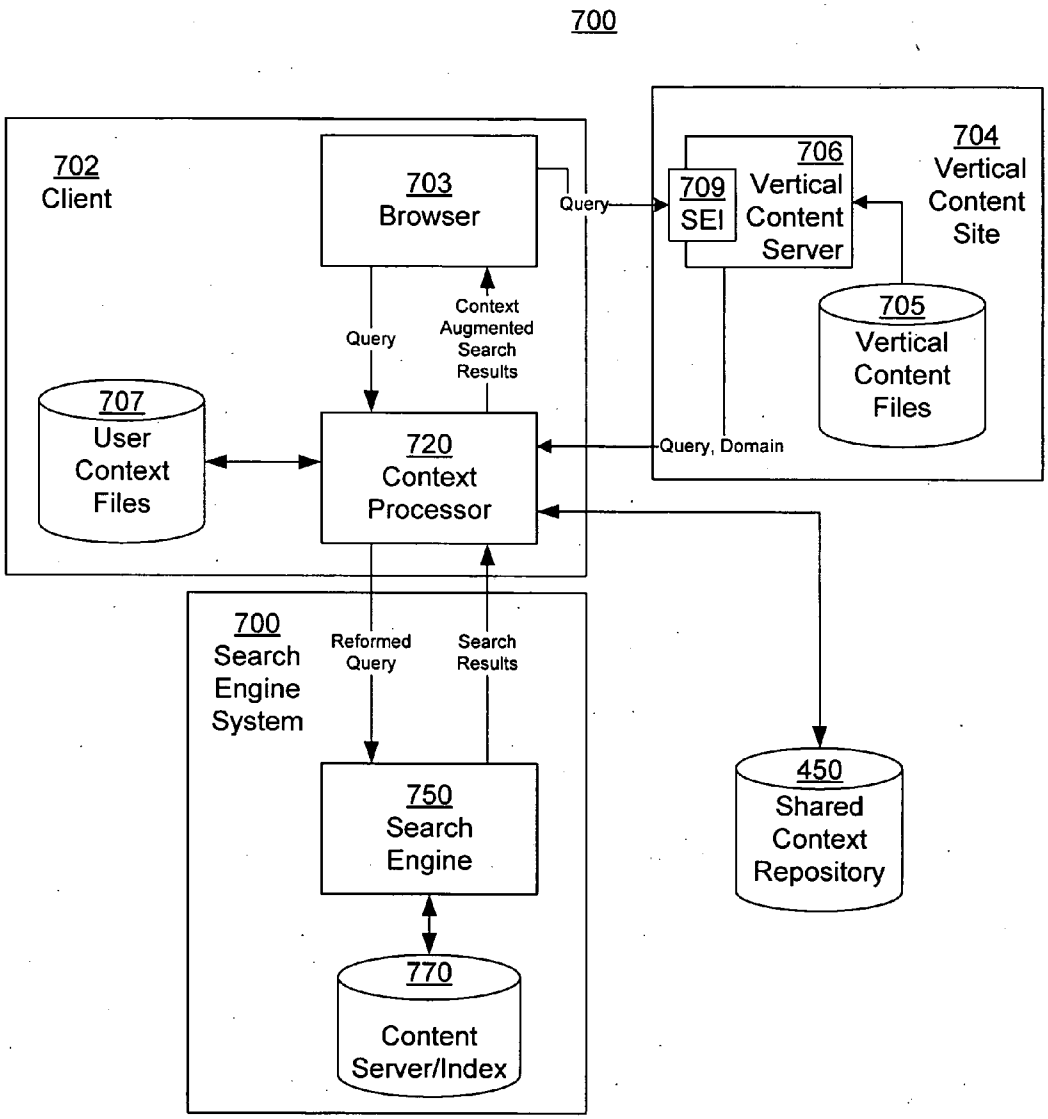

700
702
Client
703
Browser
Query
706
709
SEI
Vertical
Content
Server
704
Vertical
Content
Site
705
Vertical
Content
Files
Query
Context
Augmented
Search
Results
707
User
Context
Files
720
Context
Processor
Query, Domain
700
Search
Engine
System
Reformed
Query
Search
Results
750
Search
Engine
770
Content
Server/Index
450
Shared
Context
Repository

FIG. 7

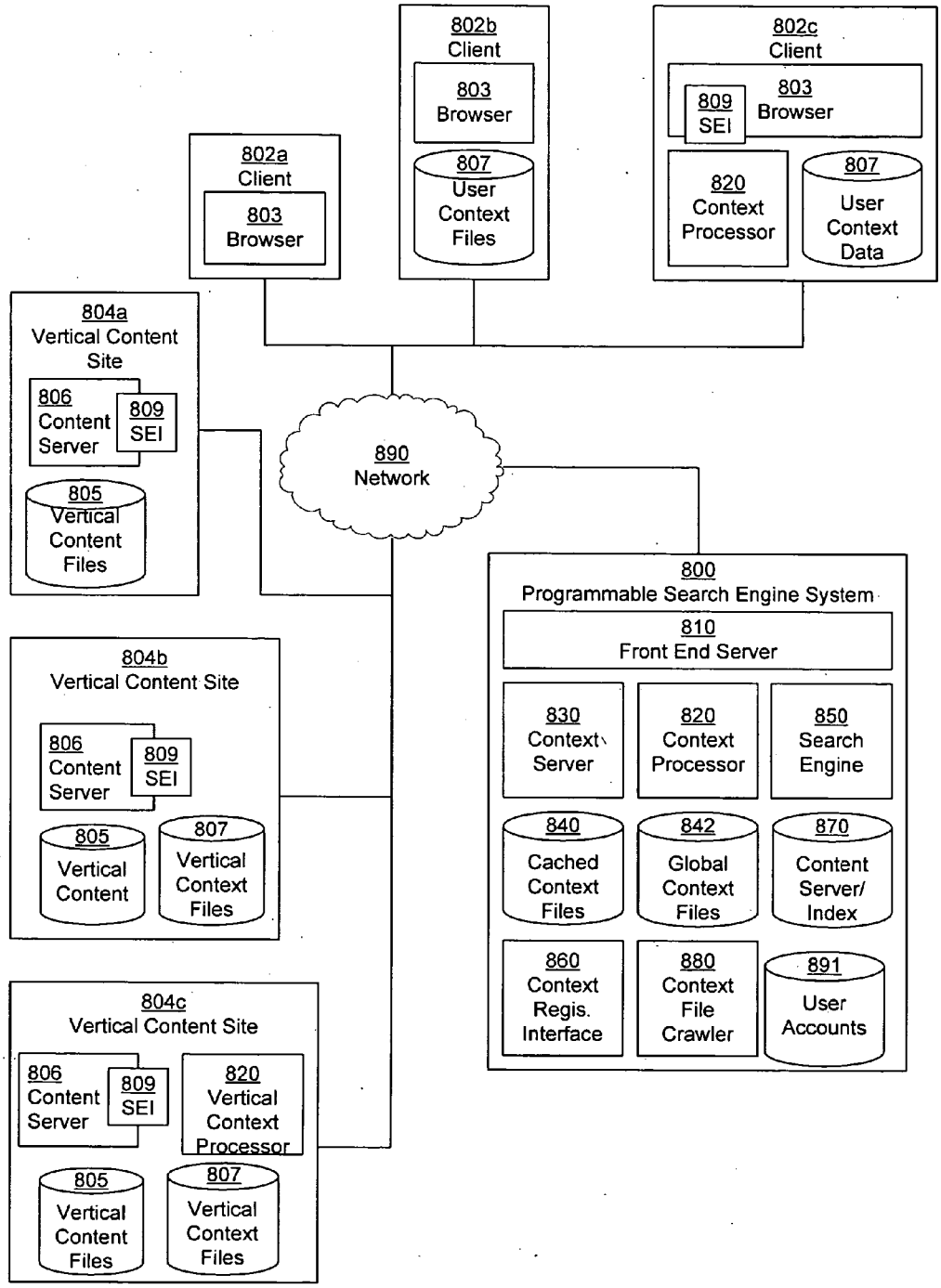

802b
Client
803
Browser
807
User
Context
Files
802c
Client
809
SEI
803
Browser
820
Context
Processor
807
User
Context
Data
802a
Client
803
Browser
804a
Vertical Content
Site
806
Content
Server
809
SEI
805
Vertical
Content
Files
890
Network
800
Programmable Search Engine System
810
Front End Server
830
Context
Server
820
Context
Processor
850
Search
Engine
840
Cached
Context
Files
842
Global
Context
Files
870
Content
Server/
Index
860
Context
Regis.
Interface
880
Context
File
Crawler
891
User
Accounts
804b
Vertical Content Site
806
Content
Server
809
SEI
805
Vertical
Content
807
Vertical
Context
Files
804c
Vertical Content Site
806
Content
Server
809
SEI
820
Vertical
Context
Processor
805
Vertical
Content
Files
807
Vertical
Context
Files

FIG. 8

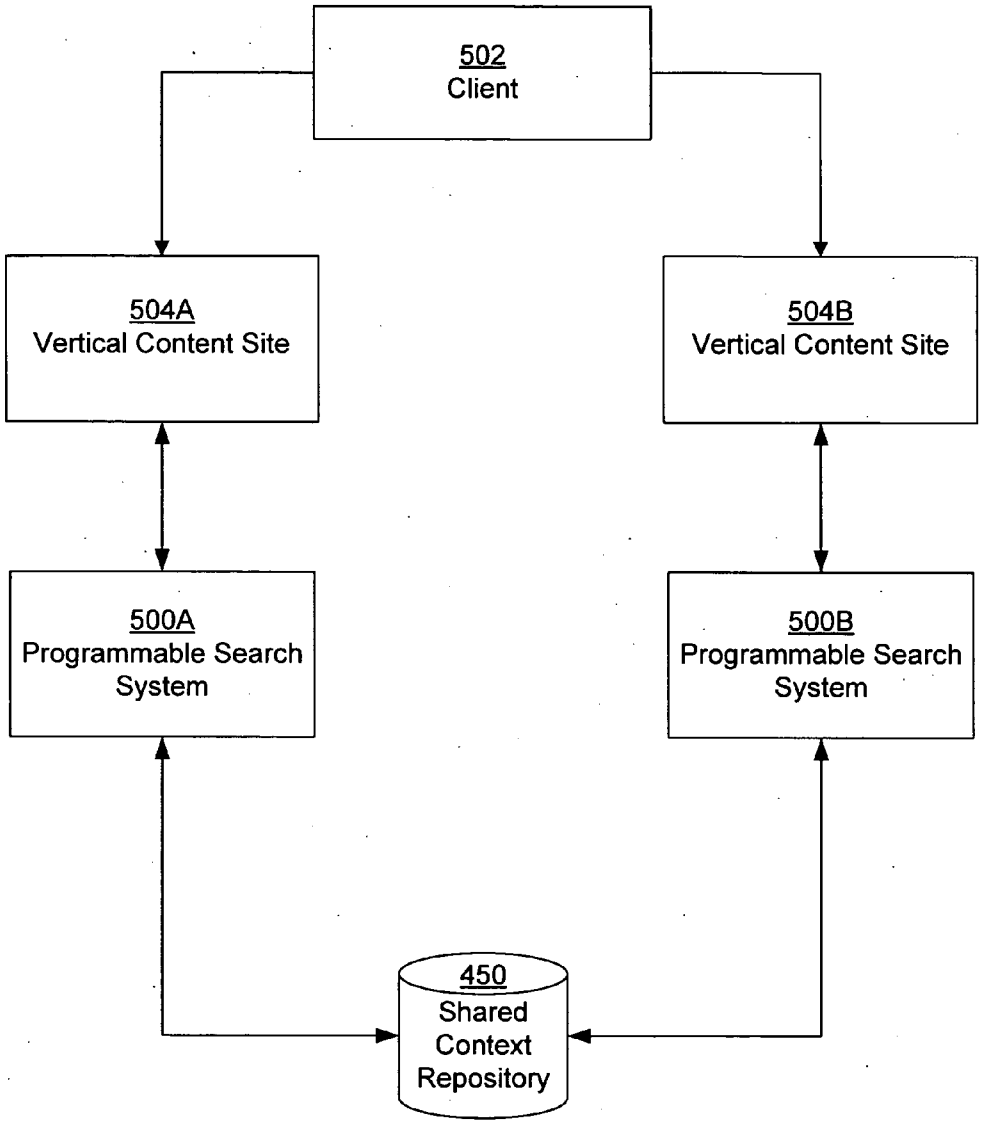

502
Client
504A
Vertical Content Site
504B
Vertical Content Site
500A
Programmable Search System
500B
Programmable Search System
450
Shared Context Repository

FIG. 9

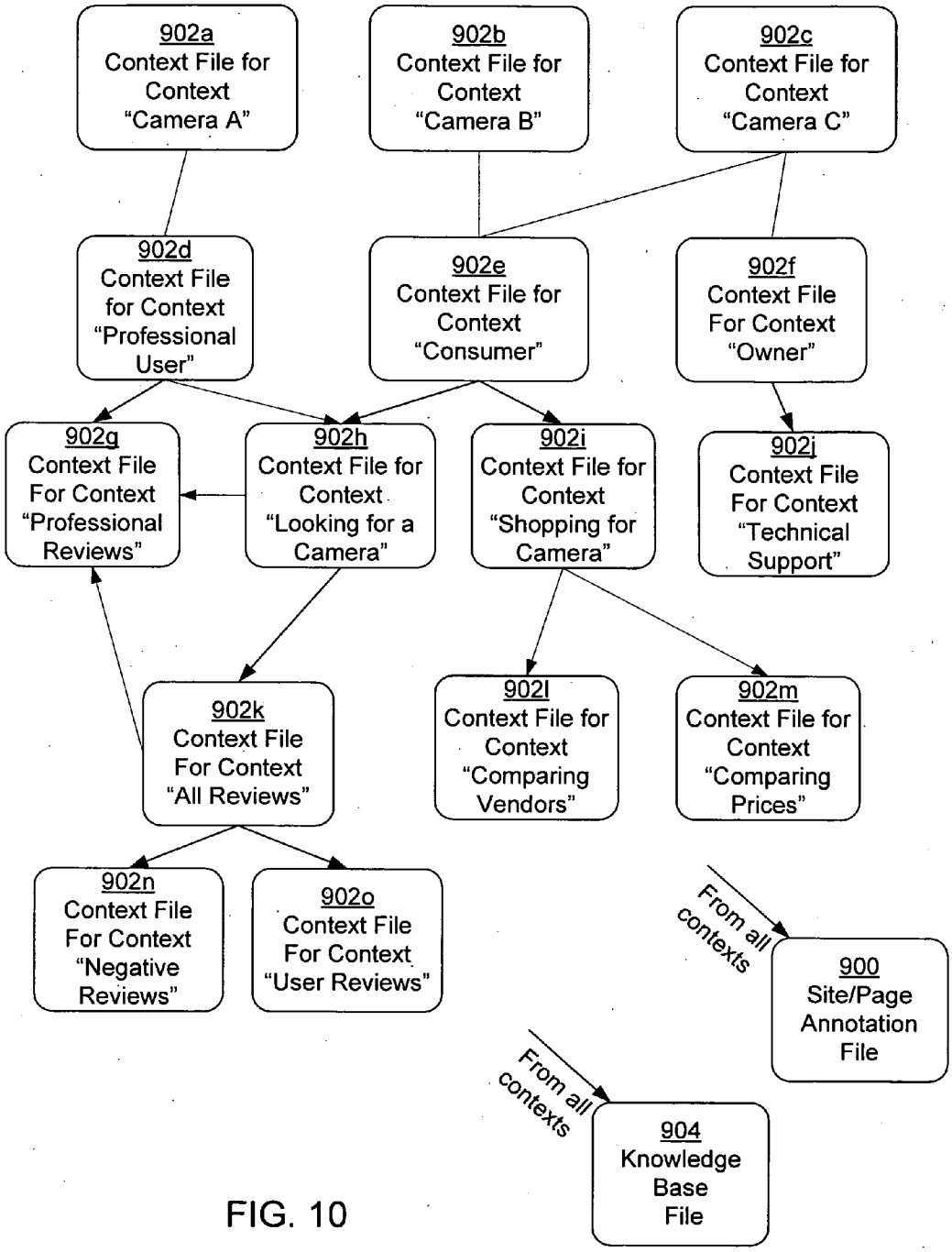

902a
Context File for Context "Camera A"
902b
Context File for Context "Camera B"
902c
Context File for Context "Camera C"
902d
Context File for Context "Professional User"
902e
Context File for Context "Consumer"
902f
Context File For Context "Owner"
902g
Context File For Context "Professional Reviews"
902h
Context File for Context "Looking for a Camera"
902i
Context File for Context "Shopping for Camera"
902j
Context File For Context "Technical Support"
902k
Context File For Context "All Reviews"
902l
Context File for Context "Comparing Vendors"
902m
Context File for Context "Comparing Prices"
902n
Context File For Context "Negative Reviews"
902o
Context File For Context "User Reviews"
From all contexts
900
Site/Page Annotation File
From all contexts
904
Knowledge Base File

FIG. 10

SHARING CONTEXT DATA ACROSS PROGRAMMABLE SEARCH ENGINES

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This invention is related to the following copending patent applications, the disclosures of which are incorporated herein by reference:

[0002] U.S. patent application Ser. No. __/___,___, filed on the same date as the present application, for "Programmable Search Engine" (attorney docket #10548);

[0003] U.S. patent application Ser. No. __/___,___, filed on the same date as the present application, for "Generating and Presenting Advertisements based on Context Data for Programmable Search Engines" (attorney docket #10549);

[0004] U.S. patent application Ser. No. __/___,___, filed on the same date as the present application, for "Aggregating Context Data for Programmable Search Engines" (attorney docket #10551); and

[0005] U.S: patent application Ser. No. __/___,___, filed on the same date as the present application, for "Detecting Spam Related and Biased Contexts for Programmable Search Engines" (attorney docket #10552).

FIELD OF INVENTION

[0006] This invention relates in general to search engines, and more particularly, to sharing context data across programmable search engines.

BACKGROUND OF INVENTION

[0007] The development of information retrieval systems has predominantly focused on improving the overall quality of the search results presented to the user. The quality of the results has typically been measured in terms of precision, recall, or other quantifiable measures of performance. Information retrieval systems, or "search engines" in the context of the Internet and World Wide Web, use a wide variety of techniques to improve the quality and usefulness of the search results. These techniques address every possible aspect of search engine design, from the basic indexing algorithms and document representation, through query analysis and modification, to relevance ranking and result presentation, methodologies too numerous to fully catalog here.

[0008] Regardless of the particular implementation technique, the fundamental architectural assumption for search engines has been that the search engine's operational model is fixed and non-alterable by entities external to the system itself. That is, the search engine operates essentially as a "black box" that receives a search query, processes the query using a preprogrammed search algorithm and relevance ranking model, and provides the search results. Even where the details of the search algorithm are publicly disclosed, the search engine itself still operates only according to this algorithm and nothing more.

[0009] An inherent problem in the design of search engines is that the relevance of search results to a particular user depends on factors that are highly dependent on the user's intent in conducting the searched (in other words, the reason they are conducting the search) as well as the user's circumstances (in other words, the facts pertaining to the user's information need). Thus, given the same query by two different users, a given set of search results can be relevant to one user and irrelevant to another, entirely because of the different intent and information needs. Most attempts at solving the problem of inferring a user's intent typically depend on relatively weak indicators, such as static user preferences, or predefined methods of query reformulation that are nothing more than educated guesses about what the user is interested in based on the query terms. Approaches such as these cannot fully capture user intent because such intent is itself highly variable and dependent on numerous situational facts that cannot be extrapolated from typical query terms.

[0010] Consider, for example a user query for "Canon Digital Rebel", which is the name of a currently popular digital camera. From the query alone it is impossible to determine the user's intent, for example, whether the user is interested in purchasing such a camera, or whether the user owns this camera already and needs technical support, or whether the user is interested in comparing the camera with competitive offerings, or whether the user is interested in learning to use this camera. That is, the user's situational facts (e.g., whether or not they own the camera currently, their level of expertise in the subject area), and their information need (e.g., the type, form, level of detail, of the request information) cannot themselves be reliably determined by either analysis of query terms, or resort to previously stored preference data about the user.

[0011] Another method of inferring intent is the tracking and analysis of prior user queries to build a model of the user's interests. Thus, some search engines store search queries by individual users, and then attempt to determine the user's interests based on frequency of key words appearing in the search queries, as well as which search results the user accesses. One problem with this approach is the assumption that queries accurately reflect a user's interests, either short term or long term. Another is that it assumes that there is a direct and identifiable relationship between a given information need, say shopping for a digital camera, and the particular query terms used to find information relevant to that need. That assumption however is incorrect, as the same query terms can be used by the same (or different users) having quite different information needs. Furthermore, such a technique is limited in its effectiveness because only one type of data (prior searches) is used. Other contextual and situational information is not captured or represented in query history and cannot therefore be used in such a methodology.

[0012] Perhaps because in part of the inability of contemporary search engines to consistently find information that satisfies the user's information need, and not merely the user's query terms, users frequently turn to websites that offer highly specialized information about particular topics. These websites are typically constructed by individuals, groups, or organizations that have expertise in the particular subject area (e.g., knowledge about digital cameras). Such sites, referred to herein as vertical content sites, often include specifically created content that provides in-depth information about the topic, as well as organized collections

of links to other related sources of information. For example, a website devoted to digital cameras typically includes product reviews, guidance on how to purchase a digital camera, as well as links to camera manufacturer's sites, price comparison engines, other sources of expert opinion and the like. In addition, the domain experts often have considerable knowledge about which other resources available on the Internet are of value and which are not. Using his or her expertise, the content developer can at best structure the site content to address the variety of different information needs of users.

[0013] However, while such vertical content sites provide extensive useful information that the user can access to address a particular current information need, the problem remains that when the user returns to a general search engine to further search for relevant information, none of the expertise provided by the vertical content site is made available to the search engine. Many vertical content sites provide a search field from which the user can access a general search engine. This field is merely used to pass a user's search query back to the general search engine. However, none of the expertise that is expressed in the vertical content site is directly available to the general search engine as part of the user's query in order to provide more meaningful search results. The expert content developer has no formal, programmatic way of passing information to the general search engine that expresses his or her expertise in their particular knowledge site.

[0014] In other words, there are no contemporary search engines that can be programmed by external entities, such as vertical content sites, during the search process itself, in way that can enhance the search process with the expertise of the content developer of the vertical content site.

[0015] Furthermore, there is generally no mechanism for vertical content sites to share information with one another regarding users and their searches. Thus, if a vertical content site contains some information about a user that might be useful in tailoring a vertical search at another vertical content site, there is generally no mechanism for transferring this information to the second site. Rather, the second site must either obtain such information independently (which can be inefficient, as well as annoying or cumbersome to the user), or provide inferior search results because the information is not made available to the second site.

SUMMARY

[0016] A user's query is processed using context information. Processing can include any combination of pre-processing operations (conducted prior to query execution) and post-processing operations (conducted on the search results from query execution). The pre-processing operations include operations to revise, modify or expand the query, to select one or more document collections on which to conduct the search, to set various search algorithm parameters for evaluating the query, or any other type of operation that can refine, improve, or otherwise enhance the quality of the user's search query. The context-processed query is then executed by a search engine to obtain a set of search results. The post-processing operations applied to the search results include operations to filter, organize, and annotate the search results as well as provide links to related contexts for other types of information or information needs. The context processing operations can be provided by a programmable search engine site, by a vertical content provider site, or by a client device. The context processing operations are controlled by context files that include commands, parameters, and instructions. The context files may be stored at the programmable search engine site, at various vertical content providers, or at a client device. Context files from multiple different sources can be used jointly. Context processing can also be limited to either pre-processing, or post-processing. The selection of which context files to apply to a given user query or a set of search results can be based on the query, the user, the client device, the vertical content site from which the query was received. The selection may be based as well on one or more subscriptions that a user has to particular vertical content providers, or popularity or reputation of a vertical content provider.

[0017] According to one aspect of the present invention, context information is shared from one vertical content site to another.

[0018] When the user issues a query associated with a particular context, either by clicking on a particular link or performing the query from a particular vertical content site, he or she is effectively communicating a piece of information about his or her current activity and/or about him- or herself. Some of this information is persistent in nature, as it refers to a characteristic of the user that will remain unchanged for some period of time; an example of such a characteristic is "vegetarian". Such information can be useful to other vertical content sites, for example to tailor results of future searches.

[0019] Accordingly, in one aspect, a repository of persistent context information for a user is stored, for example in a database. Programmable search engines associated with vertical content sites, and/or other entities, determine which types of context information are persistent and/or likely to be of use in the future. As these context information items are learned, they are stored in the repository. Stored context information can also include information entered directly by the user. For example, users can respond to online questionnaires in order to provide context information that is stored in the repository.

[0020] When a programmable search engine is running a query, it retrieves relevant context information from the repository, and uses the retrieved information for pre-processing and/or post-processing operations. This relevant context information can include information that originated from the vertical content site itself, and/or from other programmable search engines or entities.

[0021] In this manner, information obtained by one programmable search engine can be used by other programmable search engines, in order to provide more effective targeted search results for users. Furthermore, information obtained from other sources, including from the user him- or herself, can be stored and used by any number of programmable search engines. Thus, the present invention improves the level of personalization and effectiveness of searches.

[0022] The invention also has embodiments in computer program products, systems, user interfaces, and computer implemented methods for facilitating the described functions and behaviors.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] FIG. **1** illustrates a page from a host domain having a search field for accessing the programmable search engine.

[0024] FIG. **2** illustrates the results of a search from the host domain.

[0025] FIG. **3** illustrates a further page accessed from the search results page.

[0026] FIG. **4** illustrates a generalized system architecture for the programmable search engine.

[0027] FIG. **5** illustrates a first system architecture for a programmable search engine.

[0028] FIG. **6** illustrates a second system architecture for a programmable search engine.

[0029] FIG. **7** illustrates a third system architecture for a programmable search engine.

[0030] FIG. **8** illustrates a combined system architecture for a programmable search engine

[0031] FIG. **9** illustrates a system architecture wherein context information is shared among programmable search engines.

[0032] FIG. **10** illustrates an example of a set of context files.

[0033] The Figures depict various embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the illustrated and described structures, methods, and functions may be employed without departing from the principles of the invention.

DETAILED DESCRIPTION

Introduction to Programmable Search

[0034] Referring now to FIGS. **1-3**, there is shown an example of the user experience in using a programmable search system in accordance with an embodiment of the present invention. In FIG. **1** there is shown a page **100** from a host site, digitalslr.org, which is an example of a vertical content site, here the field of digital cameras. Content and organization of page **100** reflects the viewpoint and knowledge and of the entity that provides the site content. A vertical content site can be on any topic, and offer any type of information, and thus is not limited in that regard. For example, vertical content sites include sites on particular technologies or products (e.g., digital cameras or computers), political websites, blogs, community forums, news organizations, personal websites, industry associations, just to a name a few. What vertical content sites offer is a particular perspective and understanding of the world, one that may be of interest and value to some users. This perspective and understanding can be expressed, at least in part, by the content provider's organization and selection of content, as well as commentary, analysis or links to other content (e.g., commentary on other sites on the Internet). Indeed, one valuable aspect of vertical content sites is the particular collection of links to other sites that the content developer has judged to be useful in some regard, either for its depth, expertise, viewpoint, or the like. That is, users in general find value in the judgments of vertical content providers as to the usefulness of other sources of information on the Internet.

[0035] The host site includes a web server for serving pages, like page **100**, to client devices. The pages are stored in some repository, such as a database, collection of file directories, or the like. Thus, for example, the page **100** includes commentary on the latest camera offerings from various companies, as well as a link **102** to another site with relevant information about digital cameras. Of interest in this example is the search field **104**, which allows the user to search the Internet using a general search engine system (not shown), such as the Google® search engine provided by Google, Inc. of Mountain View, Calif. (of course in other embodiments, other search engines may be used). The user enters a search query in the search field **104**. Here, the query is "Nikon d100".

[0036] Activating the search button **106** causes the web server to transmit the search query to the search engine system using existing web protocols. In this example embodiment, in addition to the search query, the host site web server transmits a context file to the search engine system. Alternatively, the web server can transmit a link to the context file, or simply a context file identifier. The context file includes data that the search engine system uses to control the operation of the search engine itself in processing the search query and in presenting the search results, in effect, programming the search engine's operation. Thus, the context file, as will be further detailed below, can be understood as a set of instructions to the search engine system for processing a particular search query. The instructions can control, for example, three aspects of the search process: 1) pre-query processing operations; 2) search engine control information; 3) post-query processing operations. In addition, a context file can optionally include descriptions of (or links to) other context files, which likewise provide further programmatic control of the search engine system.

[0037] In one embodiment, the search engine also retrieves context information (if available) for the user, that was previously stored in a shared repository, such as a database. This retrieved context information provides additional guidance as to how to process the search query. In one embodiment, the retrieved context information includes characteristics of the user that were deemed persistent at the time they were stored. In one embodiment, the stored context information is associated with an expiry date, and is only used if the expiry date has not yet passed.

[0038] An advantage of the present invention is that the context information, including context information retrieved from a shared repository, provides guidance as to how to tailor search results so that the results better suit the user's needs.

[0039] In one embodiment, the search engine adds newly-learned or newly-updated context information to the shared repository. Thus, the shared repository is kept current and complete by reflecting contributions from many different search engines.

[0040] FIG. **2** illustrates an example of a search results page **200** that is provided to the user's client device following processing of the context file and the search query. This

page **200** includes a set of search results **202** that satisfy the search query, as well as additional information. First, there is displayed a name of the current context **208** that has been provided to the search engine system. In one embodiment this name is a description that the vertical content site developer has given to express the type of information need or contextual circumstances that pertains to the current search query. Here, for example, the current context **208** is for a "Camera Model", since the search query matched a specific camera model name as determined by processing of the context file. This context operates as the entry point for a user seeking information about a particular camera model.

[**0041**] Second, a number of links **204** are provided as navigational aids to further pages that address different possible information needs of the user. Each of these links **204** is associated with a related context file, which will provide further instructions to the search engine system to tailor further stages in the search process for a specific information need, and thereby construct the desired pages. For example, the first link, "If you are trying to decide which camera to buy", addresses a specific type of user information need: information about how to purchase a camera, comparisons between camera, pricing information, and the like. This need derives from a specific type of user intent, specifically the intent to purchase a camera. The second link, "Where to buy this camera from . . . ", addresses a different and more specific information need: the location of vendors for that particular camera. The last link, "If you already own one . . . ", addresses another type of information need: information that a current own would want, such as technical support and service information.

[**0042**] Page **200** also includes links **206** to other related contexts as well, such as "More Manufacturer Pages", "More Guides", "More Reviews", and so forth. These links each invoke a particular context in which the vertical content provider has characterized particular sites and pages, and then defined a filter for the search engine to select pages with the matching characteristics when processing the reformulated search query.

[**0043**] For example, the vertical content provider has here previously identified a number of different sites or pages on the Internet as being variously manufacturer sites, product review, buying guides, and so forth (e.g., according to the type of site). The vertical content provider can label (or tag) a site with any number of category labels. The labels can describe any characteristic that the vertical content provider deems of interest, including topical (e.g., cameras, medicine, sports), type (e.g., manufacturer, academic, blog, government), level of discourse (e.g., lay, expert, professional, pre-teen), quality of content (poor, good, excellent), numerical rating, and so forth. The ontology (i.e., set of labels) used by the vertical content provider can be either proprietary (e.g., internally developed) or public, or a combination thereof.

[**0044**] For example, in this example, the vertical site provider has previously identified a number of sites as containing product reviews, and has stored this information in a context file. The link **206** to "More reviews" automatically instructs the system engine system to use this context file to filter the search results during post-processing to those pages that are from sites characterized as product reviews, and satisfying the reformulated query.

[**0045**] Fourth, the page **200** includes various annotations **210** in conjunction with various ones of the search results. These annotations **210** provide the user with the viewpoint or opinion of the vertical content provider about the particular search result, as to any aspect of that search result that the provider considers significant, such as what the identified search result is about, how useful it is, or the like.

[**0046**] The placement, naming, and sequencing of the various links **204**, **206** are themselves defined in the context files. This gives the vertical content provider control over the organization and presentation of the search results, which in and of itself represents that provider's particular perspective and determination of what are the user's likely information needs, and how the search results should be organized to satisfy those needs, and which related contexts should appear in response to each level of search by the user.

[**0047**] FIG. **3** illustrates an example page **300** that is provided to the user as a result of clicking on the first link **204**, "If you are trying to decide which camera to buy." The context file associated with this link **204** is processed, and a second search is performed on the search query. This page **300** shows the context name **308**"Choosing a camera", which again reflects the selected information need of the user. The search results **302** in this context are more specifically tailored to assisting the user in evaluating digital cameras and selecting a satisfactory one. Notice, for example, the first search result is to a buying guide for digital cameras, and that there are no search results shows shown here to technical support pages.

[**0048**] Above the search results **302** are links **304** to further related contexts based on information needs, such as "Reviews, sample photographs", "Other similar cameras to consider", and "Relevant product news". Again, these links have associated context files that will control the search engine system to provide search results that are relevant to the described information needs for these contexts. Next to the search results are additional links **306**, which are also to related contexts, and for example to further professional and user reviews of digital cameras, sample photographs, and other information particularly relevant to evaluating a camera for purchase.

[**0049**] The user can thus continue to access additional related content through the various links **304**, **306**, each time obtaining search results that have been processed according to the context files associated with the selected links. In this way, the user can essentially search the Internet using the powerful capabilities of a general search engine, yet obtaining the benefit of the knowledge, expertise, and perspective of the provider of the vertical content site. Vertical content site providers benefit from this approach as it allows them to further share their knowledge and perspective with users. Vertical content providers are no longer limited to the information that they can either create themselves, provide links to, or comment upon. Furthermore, by making use of shared context data, these vertical content providers can leverage user context information that was collected by other entities, including other vertical content provides and search engines.

[**0050**] With the capabilities of the present invention, vertical content providers can define any variety of context files to meet any type of information need that users may have. The providers of the general search engine system are no

longer burdened with the task of themselves organizing and categorizing content (as is conventionally done in various directories and portals), but instead can rely upon the much deeper and vaster pool of vertical content providers—hundreds of millions or more—as compared with the limited pool of editors that may organize content directories or categorize other websites for a general search engine. The present invention thus provides any vertical content site provider with the capability to programmatically control the general search engine system on behalf of a user conducting a search.

[0051] In one embodiment, the method of the present invention is used for presenting search results generated by vertical search engines (VSEs) even when the user entered the search query at a general search site (such as google-.com). Thus, searches entered at general sites can yield results that are informed by vertical content sites. In one embodiment, each VSE is characterized by a set of query terms for which it applies. Based on these query terms and/or other factors surrounding the query and the user, the system of the present invention automatically determines how to redirect and/or process a search query, including enhancing results based on results from VSEs. Thus, the invention is able to provide improved search results that make use of context intelligence, even when the query is entered at a general search site. In this manner, the present invention integrates access to high-quality vertical search engines (and their results) into an interface for a general search engine, so as to improve the search experience even for those users who have not yet used (and may not even be aware of these vertical search engines.

[0052] For example, links to relevant VSEs can be provided on a search results page, thus providing the user with an easy way to access improved search results by simply clicking on a VSE link. Should the user do so, the query is run at the VSE corresponding to the link. In one embodiment, a recommendation and reputation network is used to select the set of VSEs presented to the user (highly-recommended VSEs are favored over less-recommended ones).

System Overview

[0053] Referring now to FIG. **9**, there is shown a generalized system architecture for sharing contexts among programmable search systems. A central shared context repository **450** contains context information for a number of users. Generally, such information includes persistent characteristics of users and/or their client machines **502**. A client **502** visits a vertical content site **504**A and enters a query. The query is processed, according to techniques described below, by a programmable search system **500**A. Context information is obtained about the user. If some or all of the context information is deemed persistent, it is stored in shared context repository **450**.

[0054] Subsequently, at some future time, if client **502** visits another vertical content site **504**B and enters a new query, the query is processed by programmable search system **500**B. System **500**B determines whether relevant context information about the user is stored in repository **450**. In one embodiment, this determination is made by searching for an identifier or index entry in repository **450** that corresponds to the user or to client **502**. If relevant context information exists, programmable search system **500**B retrieves the context information from repository **450** and uses it, according to the techniques described below, to perform pre- and post-processing on the query and results. In one embodiment, the retrieved context information is used in conjunction with context information obtained in other ways; for example, newly learned context information based on characteristics associated with the current search can be combined with context information retrieved from repository **450**.

[0055] Shared context repository **450** can be implemented using any database scheme, whether traditional or distributed, flat-file or relational. In one embodiment, shared context repository **450** is stored as a database including, for example, three fields per record: user, context and value. The value provides information for the identified context and user. For example, the value for context "owns a digital camera" could be true, false, or unknown. In one embodiment, repository **450** is indexed according to the (user, context) combination, and this index is used for retrieval.

[0056] FIGS. **4** through **8** illustrate a number of different system architectures in which the present invention can be employed. These architectures generally vary in terms of which entities provide the context files and which entities processes the context files to control the search process and search result presentation. In general, the context files can be provided by any system entity (e.g., any of a client device, a host vertical site, or the search engine system), and can likewise by processed by any system entity, or any combination there.

[0057] Referring first then to FIG. **4**, there is shown a generic system architecture. In this system architecture, there is a client device **402**, a vertical content server **406**, context server **410**, a context processor **408**, and a programmable search engine (PSE) **404**.

[0058] The client **402** can be any type of client, including any type of computer (e.g., desktop computer, workstation, notebook, mainframe, terminal, etc.), handheld device (personal digital assistant, cellular phone, etc.), or the like. The client device **402** need only have the capability to communicate over a network (e.g. Internet, telephony, LAN, WAN, or combination thereof) with the PSE **404**. Typically, a client device **402** supports a browser application, and the appropriate networking applications and components, all of which are known to those of skill in the art. The client device **402** may include as well a search engine interface that allows it to directly query the PSE **404**.

[0059] The user of the client **402** constructs and transmits a search query to the PSE **404**, via the content server **406**, which includes a search engine interface (SEI) **409**. This can be via a search query field on a host site that includes the content server **406**, along with an underlying link to initiate processing of the input text and forwarding the results thereof to the PSE **404**, as illustrated in FIG. **1**. The content server **406** selects an appropriate context file, as identified by a context ID. The selection of the context file can be based on the query itself, the client device **402**, the user identification, default selection parameters, user site behavior (e.g., page accesses, dwell times, clicks) or other information programmatically available to the content server **406**. The context ID may be a URL, a unique context name, a numerical ID, or some other form of reference to the context file.

[0060] The content server **406** transmits the query along with the context ID to the context processor **408**. Alternatively, content server **406** can provide the identified context file directly to the context processor. Depending on the embodiment, the content server **406** may also be responsible for serving content pages to the client device **402**.

[0061] The context processor **408** uses the context ID to obtain the identified context file from the content server **410**. The context processor **408** may also pass an identifier of the client device **402** (e.g., IP address, browser type, operating system, device type), the user (e.g., user ID), or host domain from which the search query is received, or the search query itself, to obtain further context files from the context server **410**.

[0062] In one embodiment, context server **410** also obtains context information from a shared context repository **450**, some of which information was collected by entities other than the context processor **408**. In one embodiment, context repository **450** is a database containing a plurality of records, each corresponding to an item of context information concerning a user or client. The database is indexed so that context processors **408** can retrieve, via context servers **410**, relevant context information for a particular query originated by a particular user. In one embodiment, more than one context repository **450** is provided.

[0063] As discussed above, a context file (or collection of context files) can include, for example, three types of programmatic information that can be used in any combination by the context processor **408** and/PSE **404** to control the search process. These are: 1) pre-query processing operations; 2) search engine parameter control; and 3) post-query processing operations. This programmatic information will be discussed as part of the operational flow.

[0064] In one embodiment, context processor **408** determines which elements, if any, of context data represents persistent information about the user and/or the client **402**. In one embodiment, such persistent information that is deemed likely to be of use to other search engines is stored in context repository **450**. Context processor **408** can store such information directly in repository **450**, or it can direct context server **410** to do so.

[0065] The context files may take various embodiments. In the some embodiments, the context files are individual files stored in a file system. In other embodiments, the context files are stored in a database system, again as either separate files, or of database entries, tables or other structures. For example, a context file in database embodiment may be stored as a collection of context records for an identified source (e.g., a specific vertical content provider), a type (e.g., knowledge base, site/page annotation, etc.), associated commands (e.g., evaluation, restriction, redirection, relation, annotation, etc.), and remaining attributes and conditions. Accordingly, no limitation is imposed on the underlying implementation of the context files by the present invention.

[0066] The context processor **408** processes the context files to perform various pre-processing operations, to programmatically generate a reformulated query. These pre-processing operations may be performed independently or in any combination to obtain a reformulated query. These include the following:

[0067] a) Query revision: the modification, addition, or deletion of or one or more terms of the original query. Such modifications include correcting spelling errors, replacing query terms, adding query terms (as conjuncts, or as disjuncts) or deleting query terms (e.g. stop word removal). The added or replaced terms may broaden or narrow the scope of a query.

[0068] b) Creation of additional queries: For example, given an original search query of "digital SLR", an additional query may be "digital camera". In one embodiment, these additional terms are incorporated into the search query as disjunctive phrases. In another embodiment, each of these additional queries is a separate query that potentially has its own filters, ranking, and the like.

[0069] These types of query reformulations are expressed in the context file as a series of query rewrite rules. The query rewrite rules generally define an output query (or query term) based on matching one or more terms of the original query (e.g., replace "digicam" with "digital camera"). Other rules may be applied automatically as defaults, without being conditioned on the terms of the query.

[0070] The second type of control information processed by the context processor **408** are search engine control data. These include:

[0071] a) selection of one or more search engines for processing the reformulated search query. The PSE **404** may include any number of different search engines, each of which is optimized for certain types of searches. For example, different search engines are typically used for text searches, image searches, and audio searches. A search engine typically will generate an information retrieval score for various documents in terms of their relevance to the search query. A context file can specify which search engine or engines is/are to be used (e.g., by identification of particular URLs for the search engines). A single search can integrate results from different engines. The context processor **408** extracts the identified search engine(s), and constructs the appropriate query string using the reformulated query.

[0072] b) selection of one or more document collections on which to search. A search engine system will typically have access to multiple different document collections, which can be searched jointly, or individually. The provider of the context file may instruct the PSE **404** to use one or more specific document collections for a particular search. For example, a vertical content site for healthcare professional, may receive a search for "migraine", and instruct the search engine system to search the PubMed database provided by the National Library of Medicine, rather than a more general search of the Internet. This constraint better tailors the results to the medical literature most likely to be relevant to the information need of a healthcare professional, rather than the typical results to such a query on the Internet. The context file can specify which document collections are to be used (e.g., by specification of a database, index, or other context repository). The context processor **408** extracts this information from the context file as well, and passes it the selected search engine as a parameter.

[0073] c) specification of search engine parameters for use during query processing. Most search engine algorithms operate under a large number of parameterized controls

when generating information retrieval scores, such as threshold values for scoring query term matches, iteration cycles, waiting of links, terms and other query or document attributes. Normally, these parameters are not accessible to entities outside of the search engine system, but rather are fixed by the search engine provider. However, in some embodiments of the present invention, the search engine system may be configured to receive and use any of these types of parameters, thereby giving further incremental programmatic control of the search engine to the vertical content developments. Again, the context processor **408** extracts these parameters from the context file and passes them to the search engine **404** as parameters.

[**0074**] The context-processed query, which includes the reformulated query and the search engine control data (if any) that are specified in the context file, is thus provided to the PSE **404**. If multiple queries are constructed during pre-processing, the context processor sends each of the multiple queries and their associated search engine control data (which may be individually varied) for each additional query.

[**0075**] The PSE **404** processes the reformulated query using the search engine control data (if any) to obtain a set of context-processed search results, and provides these search results to the context processor **408**. If multiple queries are processed, then the PSE **404** can merge the results from these searches.

[**0076**] The context processor **408** then provides various post-processing operations, which again may be performed independently or conjointly. The results of this post-processing made part of the context-processed search results. The post-processing operations include:

[**0077**] a) filtering the context-processed search results using filters specified in the identified context. The context file may specify one or more filters that the context processor **408** can apply to further limit the documents that are included in the search results. These filters are expressed in terms of rules that match metadata with particular metadata associated each search result. The metadata can include both native metadata to the document, such the document type, date, author, site, size, or labeled metadata associated with the document, that is the labeled characteristics provided by the vertical content provider (or others).

[**0078**] For example, the filters may be defined to exclude documents of certain types (either physical types, e.g., image files, or logical types, e.g., "reviews"), from particular sites or internet domains (e.g., documents from the .biz or .gov domain), websites, or of a certain vintage (e.g., documents published before Mar. 3, 2005). Referring back then to the example of FIG. **3**, the link **306** for "More Professional reviews" would invoke a filters defined to select only documents labeled as "professional", "product reviews". Again, these labels can be provided by the vertical content provider from which the original query was sourced, or from some other source. These options will be more fully discussed below.

[**0079**] b) ranking of the context-processed search results using ranking parameters specified in the context file. The PSE **404** includes a ranking function that ranks the search results based on the respective information retrieval scores. The context file can include ranking parameters, such as weighting factors to increase or decreases the IR scores for particular types of documents, for documents from selected sources. The ranking function may also operate on identifiable native or labeled metadata. For example, the rankings can be adjusted based on length of document, publication date, or document format just to name a few. Alternatively, the ranking may be adjusted based on labeled metadata, such ranking by expressed "rank" value, or by as increasing the native ranking of documents labeled as "expert" by a weight factor, or increasing the ranking of documents having some specified quality measure of "10". The context processor **408** can use these ranking parameters to rank the documents in the search results.

[**0080**] c) clustering of the search results using clustering parameters. The context processor **408** may also cluster (group) the search results according to parameters provided in the context file. The parameters can specific clustering based on native or labeled metadata. Thus, all documents labeled as "professional reviews" can be clustered together; or all documents where are image files can be clustered, or documents from a given domain (e.g., all documents from xxxx.com).

[**0081**] d) providing navigational links in the context-processed search results to additional contexts. As illustrated in FIGS. **2** and **3**, the context processor may also provide links that can be accessed to invoke additional searches for further refinements of the information needs of the user. Each such related context link invokes another cycle of pre-processing and/or post-processing by the context processor **408** and if so instructed, another cycle of query processing by the PSE **404**.

[**0082**] e) annotating the context-processed search results using annotations specified in the identified context. As illustrated in FIGS. **2** and **3**, the context file may also provide specific annotations **210** that can be included with any of the search results.

[**0083**] In one embodiment, the system of the present invention does not change the order in which the initial results are presented, but annotates the results with the labels that apply to them. Clicking on a label issues a new search restricted to the results matching this tag. In yet another embodiment, these annotations need not be labels but can be links to relevant pages on other sites.

[**0084**] Thus, the context files can include conditional instructions that define various types of Annotations. These annotations are provided by the annotate command. In one embodiment, this command has the following syntax:

```
<Annotate count="n">
    annotation condition*
    annotation action*
</Annotation>
```

[**0085**] The annotation condition operates in a similar manner to a restriction condition. Here, the annotation condition is evaluated with respect to the attributes (tags), if any, associated with the search results, as compared to the entries in the site/page annotation file. Any attribute (or set of attributes) can be used as annotation conditions, such as the type, source, year, location, or the like, of a document or

page. The context processor receives the search results from the search engine, and compares each result (be it a site, page, media page, document, etc.) with the entries listed in the site/page annotation file **900**. Results that satisfy the condition are annotated with the annotation action. Annotate commands can be used by themselves or in combination with any of the other commands, including Restrictions.

[**0086**] In yet another embodiment, the query does not originate at the vertical content site, but at a general search engine site. The system of the present invention provides a mechanism by which the knowledge provided by the vertical content site is applied even for searches entered at a general site such as google.com. In one embodiment, the user indicates to the search engine, either while using the VSE or through a sign up process similar to that used to subscribe to RSS feeds, that he or she would like to apply the VSE's contexts which conducting searches of a particular type. In another embodiment, selection and use of a particular VSE is performed automatically.

[**0087**] The context processor **408** then provides the context-processed search results to the client device **402**. As noted, the user can access any of the related context links, or perform entirely new queries, again making use of any context files that are selected based on such queries.

[**0088**] The client device **402** may also query the PSE **404** directly, either through its search engine interface **409**, or simply by going to the website of the PSE **404** entering the query directly there. In this scenario, context processing is still handled by the context processor **408** in manner described above.

[**0089**] Any of the above-described pre-processing and post-processing operations can be applied using a context obtained from shared context repository **450**, or a context identified by content server **406** based on known information about the user or client **402**, or any combination thereof.

[**0090**] Referring now to FIG. **5**, there is a shown a system architecture in which the context processing operations are provided by the PSE system itself. Again, there is a client device **502** as before, including a browser **503**, along with a host vertical content site **504**, and a PSE system **500**. The vertical host vertical content site **504** includes a vertical content server **506** (e.g., a web and/or application server) and vertical content files **505** (e.g., a database or directory of web pages). Also present are vertical context files **507**. The vertical content site **504** also includes a search engine interface **509** to the PSE system **500**, such as a search field and search button as illustrated in FIG. **1**. The user accesses the vertical content site **504**. From that site, he or she enters a search query to be processed by the PSE system **500**. The vertical content server **506** processes the search query to determine a context ID for an appropriate context file, and transmits the search query and context ID to the PSE system **500**. For example, the context ID can be transmitted as a parameter in a URL to the PSE system **500**. The vertical content site **504** also includes a number of conventional components (e.g. firewalls, router, load balancers, etc.) not shown here in order to not obscure the relevant details of the embodiment.

[**0091**] The PSE system **500** includes a number of components. A front end server **510** provides the basic interface for receiving search queries. The front end server **510** extracts the context ID and query, and passes that to a context processor **520**. The front end server **552** may also provide an identifier of the client device or the user to the context processor **520**. The context processor **520** provides the context ID and query, to the context server **530**. The context server **530** uses the context ID to retrieve a context file from a repository of cached context files **540**. The context files are received from any vertical content site **504**, via a registration interface **560**. This allows any provider of a vertical content site **504** to define the context files that are to be used for handling queries from their site and upload such context files for storage by the PSE system **500**. Alternatively, the context files are extracted from the vertical content sites **504** by a context file web crawler **580**. The registration and crawling methods may be used together. One implementation would be for the vertical content site **504** to first register its context files **507**, which includes putting the site address on a crawl list. Subsequently, the crawler **580** crawls the site **504** to obtain any updates to the context files **507**. Caching of the context files ensures very high speed processing of the context files at query time, since context processor **520** does not need to retrieve the context files from the remotely vertical content site **504**, and thereby does not incur network latency (or problems with the vertical content site being unavailable).

[**0092**] The context server **530** may also obtain context files from a repository of global context files **542**. These context files can be derived from data mining on the cached context files **540**, provided by the provider of the PSE system **500**, or any combination thereof. The context server **530** may also obtain context data from a shared context repository **450** that includes information that was collected by entities other than the context processor **520**. Such context data can include any information that is deemed relevant and persistent with respect to the user and/or client **502**.

[**0093**] The context server **530** provides the retrieved context file(s) to the context processor **520**. The context processor **520** performs the appropriate preprocessing operations (if any) as defined in the context file to generate the reformulated query, and establish the search engine control data as set forth above, as part of the context-processed query. The search engine **550** receives the contextprocessed query, including reformulated query and search engine control data, and executes a search on same to provide a set of context-processed search query results. These results are passed back to the context processor **520**, which performs the post-processing operations on the search results as defined in the context file, to further modify the context-processed search results. These contextprocessed results are then transmitted back to the client device **502**.

[**0094**] This architecture provides various benefits. First, it provides for high-speed access to the context files and eliminates reliance on the availability of the remote vertical content sites to serve their context files on demand.

[**0095**] Second, collection and aggregation of the context files allows for various systemic benefits to be achieved from analysis of the context files.

[**0096**] Specifically, the following types of information may be aggregated from the collected context files. The rules used to define the query preprocessing operations can be accumulated and used to identify the most frequently used

rules for various query terms. To a large extent this type of information is more reliable, having been essentiality voted on by a large population of interested providers, as opposed to rules designed by a very small team of editors.

[0097] Similarly, analysis of the search engine control yields identification of most frequently used search engines, indices, and parameters for particular queries or types of queries. Analysis of the query post-processing operations also identifies the most frequently used annotations, related contexts, ranking and filtering operations.

[0098] As mentioned above the context files includes label metadata used by the vertical content providers to describe the characteristics of any site or page on the Internet. In one embodiment, these labels are selected from a publicly provided ontology, so that vertical content providers use the same set of labels to characterize the content of the Internet. The ontology of labels can describe categories and instances of any type. The ontology includes, for example, topics, information types, information sources, user types, and rating scales, just to name a few possible aspects of the ontology. Accordingly, from the cached context files **540** a categorization of Internet content can be derived and validated. By way of simple example, all Internet sites labeled as type "buying guide" and category "digital camera" can be extracted from the cached context files **540**. A directory of these digital camera buying guides can then be constructed, for example by selecting those sites having that have a minimum number of appearances in the context files. This approach again leverages the collective judgment of the vertical content providers—that is, the wisdom of crowds—as to the nature, type, and quality of content on the Internet.

[0099] From the foregoing, the PSE system **504** can extract and establish a collection of globally optimized context files, where the query pre-processing rules, search engine control data, and query post-processing rules are derived from statistically analysis of cached context files for the frequency, distribution, variability and other measures of the usage of context information. Such globally optimized context files can be processed using user-specific information that is deemed persistent and relevant; such information can be stored in shared content repository **450** for use by other search engines.

[0100] One scenario for this architecture is to support direct search queries with post-query context processing. In this embodiment, a user query is received directly from the client device **502**, without first being passed through a vertical content provider site **504**. The user's search query can be received directly at the website of the PSE system **500** (e.g., via search query page), or a search interface in browser toolbar, application, or system extension (e.g., a search interface on the user's desktop). In any event, the user's search query is handled without context based preprocessing (that is, query modification based on a vertical content provider's context files), though internal adjustment of the search query may be performed as part of native search operations. The search results are then post-processed with one or more context files, to provide the various types of navigational links, related context links, and/or annotations on search results as described and illustrated in FIGS. **2** and **3**.

[0101] Another beneficial aspect of this architecture is that analysis of the context files also allows for integration of advertisement purchases based on contexts. That is, advertisers can bid for placement of their advertisements in specific contexts, rather than by specific query terms. For example, an advertiser may bid for placement of an advertisement for its digital camera when the context file for a query indicates that the user is shopping for a particular camera model, but not when the user is seeking technical support. This allows advertisers to more precisely focus their advertising efforts based on the user's information needs—which have been expressly described by the context files, rather than merely inferred from the query terms.

[0102] Referring now to FIG. **6**, there is shown an embodiment of a system architecture in which the context processing is provided by the vertical content site itself. In this embodiment again there is a client device **602** including a browser **603**, along with a host vertical content site **604**, and a general search engine system **600**. The vertical content site **604** includes a vertical content server **606** and vertical content files **605** (e.g., a database or directory of web pages). The vertical content site **606** also includes a search engine interface **609** to the search engine system **600**, such as a search field and search button as illustrated in FIG. **1**. The user accesses the vertical content site **604** and from that site can enter a search query to be processed by the search engine system **600**.

[0103] In this embodiment, the vertical content site **604** also includes various components for context processing. First, the vertical content site **604** includes a vertical context processor **620** and local vertical context files **607**. As before, vertical content server **606** receives a search query from the client device **602**, e.g., via the browser **603**, and processes the search query to determine a context ID for an appropriate context file. This information is now provided to the vertical context processor **620**. The context processor **620** passes the context ID (and optionally the client device ID, user ID, and query) to the context server **630**. The context server **630** uses the context ID to retrieve a context file from the vertical context files **607**.

[0104] In one embodiment, the context server **630** also obtains context information from a shared context repository **450**, some of which information was collected by entities other than the vertical context processor **620**. In one embodiment, context repository **450** is a database containing a plurality of records, each corresponding to an item of context information concerning a user or client. The database is indexed so that vertical context processors **620** can retrieve, via vertical context servers **630**, relevant context information for a particular query originated by a particular user. In one embodiment, more than one context repository **450** is provided.

[0105] The context server **630** provides the retrieved context file(s) to the context processor **620**. The context processor **620** performs the appropriate preprocessing operations as defined in the context file to generate the contextprocessed search query (including the search engine control data as set forth above). The vertical context processor **620** then invokes the search engine **650** to process the context-processed query.

[0106] The search engine **650** receives the reformulated query and search engine control data, and executes the search accordingly, generating the contextprocessed search results. These results are passed back to the context proces-

sor **620**, which performs the post-processing operations on the search results as defined in the context file, to further modify the context-processed search results. These processed results are then transmitted back to the client device **602**.

[**0107**] The context processor **620** may also provide some or all of the search engine control data to the search engine, depending whether the search engine **650** exposes an application programming interface. In some embodiment, where the search engine **650** is closed, then the context processor **620** simply passes the queries to the search engine **650** and operates on the results. In this embodiment, the context processor **620** itself would use at least some of the search engine control data, for example, selection of which search engine to use. This gives the vertical content site provider control as to which search engines **650** to use with which types of user queries.

[**0108**] In one embodiment, vertical context processor **620** determines which elements, if any, of context data represents persistent information about the user and/or the client **602**. In one embodiment, such persistent information that is deemed likely to be of use to other search engines is stored in context repository **450**. Vertical context processor **620** can store such information directly in repository **450**, or it can direct vertical context server **630** to do so.

[**0109**] Referring now to FIG. **7**, there is shown an embodiment of a system architecture in which the context processing is provided by the client device site. In this embodiment again there is a client device **702** including a browser **703**, along with a host vertical content site **704**, and a general search engine system **700**.

[**0110**] As before, the vertical host vertical content site **704** includes a vertical content server **706** and vertical content files **705** (e.g., a database or directory of web pages). The vertical content site **706** also includes a search engine interface **709** to the search engine system **700**, such as a search field and search button as illustrated in FIG. **1**. The user accesses the vertical content site **704** using the browser **703** and from that site can enter a search query to be processed by the search engine system **700**.

[**0111**] In this embodiment, the client device **702** includes the various components for context processing. First, the client device **702** includes a browser **703**, for accessing the vertical content site **704** as well as any other available site on the network. The client **702** includes a vertical context processor **720**, which can operate a plug-in to the browser **703**, or Java applet. Once the user makes the query via the vertical content server **706**, that query is also provided to the vertical context processor **720**. The context processor **720** again processes the search query to determine a context ID for an appropriate context file. Since the operation is local to the browser, the context processor **720** can use the context ID to retrieve a context file from the user context files **707**.

[**0112**] In one embodiment, vertical context processor **720** also obtains context information from a shared context repository **450**, some of which information was collected by entities other than the vertical context processor **720**. In one embodiment, context repository **450** is a database containing a plurality of records, each corresponding to an item of context information concerning a user or client. The database is indexed so that vertical context processors **720** can retrieve relevant context information for a particular query originated by a particular user. In one embodiment, more than one context repository **450** is provided.

[**0113**] The context processor **720** then performs the appropriate preprocessing operations as defined in the context file to generate the contextprocessed query. The vertical context processor **720** then invokes the search engine **750** to process the context processes query. The search engine **750** receives the context-processed query, and retrieves search results, forming the contextprocessed results. These results are passed back to the context processor **720**, which performs the post-processing operations on the search results as defined in the context file, to further modify the context-processed search results. These results are then passed back to the browser **702**.

[**0114**] In one embodiment, vertical context processor **720** determines which elements, if any, of context data represents persistent information about the user and/or the client **702**. In one embodiment, such persistent information that is deemed likely to be of use to other search engines is stored in context repository **450**.

[**0115**] An advantage of this architecture is that it allows the user to establish and user their own context files. Just as individual vertical content providers have their individual expertise and viewpoint, so to do individual users. Thus, a user may define context files to categorize and label particular websites, for example, identifying the site that she considers most authoritative or useful for particular topics. The user can also define query pre-processing operations, or more likely import such operations from others (e.g., experts in various topical domains) who publish context files for this purpose. Similarly, the user can define post-processing operations that allow for customization in the presentation of results, including arrangement of results into clusters or grouping that the user feels most comfortable with. For example, a user can define a personal context file in which search results are always clustered into academic (.edu), government (.gov), retail shopping (sites having metadata or text indicative of online purchasing), and image files.

[**0116**] The architectures illustrated in FIGS. **4-7** can all operate concurrently with different types of the individual systems operating together. FIG. **8** illustrates this system architecture for mutual and concurrent context processing. All of the system elements communicate via a network **890**, such as the Internet.

[**0117**] First, the PSE system **800** includes a complete set of components as described with respect to FIG. **4**. The operative features of these components have been previously described and so are not repeated here.

[**0118**] Next, three types of client devices **802** are in operation. Client device **802***a* simply has a browser **803** by which it accesses various sites on the Internet. Client device **802***b* includes a browser **803**, as well as user context files **807**, which can be passed to any available context processor **820** for processing in conjunction with a search query provided by the user.

[**0119**] Client device **802***c* includes a browser **803** and user context files **807**, as well as its own context processor **820**. This enables the client **802***c* to perform local context processing on the user's search query prior to sending the query to the search engine, and to perform post-processing opera-

tions after receiving the search results. This client's browser **803** also includes a search engine interface **809**, enabling direct querying of the PSE system **800**. Other clients **802***a* and **802***b* may also include search engine interfaces **809**, for example, in the toolbar of their respective browsers **803**.

[**0120**] The three types of different vertical content sites **804** are also shown. Vertical content site **804***a* includes a content server **806**, along with a search engine interface **809** to the PSE system **800**, as previously described. The server forwards a user's query (from any type of the client devices **802**) to the PSE system **800**, providing as well the context ID associated with the user's current context (along with any context related information received from the client device). The site does not need to store its own context files, as these can be stored at the PSE system **800** in the cached context file database **840**.

[**0121**] For this type of vertical content site **804***a*, the PSE system **800** provides all of the context processing operations. Here, the site **804***a* does not provide any specific context ID information. As a result, the PSE system **800** can provide its own context identification mechanisms, for example based on the site **804***a*, the client **802**, the query terms, or the like. Using the context information, the context server **830** retrieves the appropriate global context files **842**, and the context processor **820** uses these files for the context processing operations, including pre-processing of the search query, control of the search engine operation and parameters, and post-query processing. The programmable search engine site **800** passes the context-processed search results back to the requesting client, either directly, or within the scope of the vertical content site **804***b*, e.g., using framing techniques.

[**0122**] As with vertical content site **804***a*, vertical content site **804***c* includes its own content server **806** search engine interface **809**, vertical content files **805**, as well as local vertical context files **807**. This site **804***b* receives a search query from a client device **802**, and forwards the query along with the context ID for the query context to the PSE system **800**. The site's vertical context files **807** are cached in the PSE system's cached context files **840**. The PSE system **800** receives the context ID, and uses its context server **830** to retrieve the associated context files for site **804***b* from the cached context files **840**. The context server **830** may also retrieve any applicable global context file **842**. The PSE context processor **830** then processes the retrieved context files, generates the context-processed search query and processes the queries via the search engine **850**. The context-processed search results are then further post-processed by the PSE context processor **820**, again in accordance with either the site's context files or the global context files **842** (including where appropriate a combination thereof).

[**0123**] The last type of vertical content site **802***c* includes its own content server **806** search engine interface **809**, vertical content files **805**, local vertical context files **807**, as well as a local, vertical context processor **820**. The local context processor **820** receives the user's search query, along with the context ID for the user's context, and using the referenced context files performs the appropriate pre-processing operations on the query prior to transmitting it to the PSE system **800**, along with the search engine control data specified by the context files.

[**0124**] Here, the PSE system **100** can provide various levels of services to the vertical content site **804***c*. Minimally, the programmable search engine system **800** can process the received context-processed queries, and execute these queries accordingly via the search engine **850**, providing the context-processed search results back to the local context processor **820** for further modification. The local context processor **820** for the vertical content site **804***c* provides further post-processing operations specified by the identified context, and then forwards the final set of context-processed search results to the client device **802**.

[**0125**] Alternatively, the PSE system **800** can perform some specific context processing operations as instructed by the local context server **820**, whether pre-processing, or post-processing, or control of the search engine operations. For example, the local context processor **820** may perform the pre-processing operations to reform the queries, but then use the search engine control data to specify which document collections and search algorithms the search engine **850** should use.

[**0126**] In addition, the PSE system **800** may add its own layer of context processing based on its global context files **842**, including generation of additional reformulated queries, control of the search engine **850**, and post-processing of search results prior to returning them to the vertical content site's local context processor **820**. The vertical content site **804***c* can forward the context-processed search results to the client device **802** directly, or can invoke another layer of post-processing operations by the local context processor **820**, perhaps to further fine tune the organization, commenting, or navigation features thereof.

[**0127**] The PSE system **800** can provide context processing directly to user queries input at the PSE site from any of the client devices **802**. The user's search query can be received directly at the website of the PSE system **800** (e.g., via search query page), or a search interface in browser toolbar, application, or system extension (e.g., a search interface on the user's desktop). Since the user's query is not coming from a vertical content provider, the PSE system **800**'s context processing can use the global context files **842**, including those for annotating search results with links to potentially useful context for the user.

[**0128**] The degree of context processing for direct queries can be varied to include either pre-processing or post-processing individually, or a combination of both. One embodiment of direct query handling provides a context-based post-processing on the search results without context based pre-processing (e.g., query modification). Here, the user's search is received and executed without pre-processing based on the context files of a specific vertical content provider (though some internal adjustment of the query and selection of search indices may optionally be employed to provide the most relevant search results). As described with respect to FIG. **5**, the search results are then post-processed with one or more context files to provide the various types of navigational links, related context links, and/or annotations on search results as described and illustrated in FIGS. **2** and **3**.

[**0129**] The post-processing operations in this scenario can use either global context files **842**, or can be based on the context files of any number or selection of the vertical content providers. In one embodiment, a user can identify which the vertical content provider whose context files are to be used for context processing. Identification can be done

via a subscription model, in which the user subscribes to have such context processing done for her or her queries, for example via a subscription interface (e.g., page) at the website of the vertical content provider, which then forwards an identifier of the user or the user's client device to the PSE **800**. A user may subscribe to a particular vertical content provider in order to have that provider's expertise, perspective or viewpoint applied to the user's search queries and results, without the user having to always enter a query from that vertical content provider's site.

[**0130**] For this embodiment, the PSE system **800** includes a user account database **891**, which stores for each user various types of personal preferences for searches, including the subscriptions to particular vertical content providers. The PSE **800** also provides a registration interface (allowing the user to register with the PSE system **800** for storing search preferences, subscription information, and other user settings), and a login interface for the user to login and have the user's settings applied to the user's queries. Direct queries received from the user and/or the user's client device **802** are identified by the PSE **800** and then the appropriate context files to which the user subscribed are used for context processing. In another embodiment, subscription-based context processing is provided for direct user queries for both pre-processing and post-processing operations.

[**0131**] The selection of which vertical content provider's context files are to be used (whether for pre-processing, post-processing or both) can be based on other factors beyond a user's subscriptions, as some users may not have subscribed to any particular vertical content provider. In one embodiment, the selection is based on a popularity measure for each vertical content provider whose context files are included in the cached repository. The popularity measure can be based on web access statistics, like number of unique visitors to a vertical content provider's site each month (or other time period), number of hits to such site, number of current subscribers to the vertical content provider. These and other statistical measures can be combined into a popularity measure. Alternatively, or additionally, the selection can be based on a reputation measure (or rank), where the reputation of each vertical content provider is judged and rated by users.

[**0132**] In summary, the foregoing provides a general overview of the operations and various system architectures useful with the present invention. As can be seen, the present invention can be practiced in a number of different and complementary embodiments. The capability of the present invention enable any system entity to provide context files, context processing, or both, results in both tremendous flexibility and power. The flexibility allows for rapid, widespread and easy implementation of the present invention. Any system entity can provide various levels of operative support, and cooperate with any other system entity, according to the techniques described herein.

[**0133**] The context files and context processing capability can be readily implemented in any vertical content site and in any client. The power of the system derives in part from such widespread distribution and implementation: the more context files and context processing is adopted, the more contextual information can be accumulated and leveraged, for example in the global context files. This enables the PSE system to continually refine and adapt its capabilities to the information needs of the wide variety of users. Further, the widespread use of context files by vertical content developers continually expands the range of information needs and perspectives that can be satisfied, as well as the depth and quality of that information that is used to satisfy such needs.

[**0134**] Referring now to FIG. **10**, there is shown an example of a set of context files as might be developed by a vertical content provider for a digital camera related website. This simplified example is used only to illustrate some of the basic aspects of context files, and not as definitive statement of their characteristics.

[**0135**] In this example, the vertical content provider has provided a variety of context files that suit different types of information needs, and different types of available resources. Context files **902** are illustrative of contexts defined for various types of users of digital cameras, such as a professional user searching for a digital camera, a consumer searching for a digital camera, and an owner who already has such a camera. Each of these types of users has different information needs and typically different approaches to evaluating the information she obtains. For example, a professional user is typically most concerned with technical performance issues such as picture quality, durability, and compatibility with an existing set of professional equipment, whereas a consumer user is typically concerned with ease of use, convenience and price. Both of these types of users are seeking information during their purchase process that is quite different from an existing owner. An owner is not typically interested in obtaining further opinions or evaluations of a product, but rather information pertaining to its use, technical support, service, or warranty issues.

[**0136**] Each of these three user type context files **902** contain instructions that enable a context processor to respond to a specific query according to the expected information needs of the user. Thus, the context file **902***d* for the professional user may include query revision rules to modify a received query such as "Nikon camera" to "Nikon DX2", which is a current model of a professional digital SLR, and one deemed by the content provider to be of most interest to the professional user. By contrast, the context file **902***e* for the consumer user may include query revision rules to modify this same query to "Nikon Coolpix 7600", again a current model of the Nikon cameras, and determined by the content provider to be the best Nikon camera for a typical consumer user. Continuing this example then, the vertical content site would pass the consumer context file **902***e* to a context processor along with the user query of "Nikon camera", and the context processor would use the query modification rules to generate the appropriate revised query for execution.

[**0137**] The arrangement and interrelationship of the context files is highly flexible and is decided by the particular vertical content provider. Each of the context files **902** can point to any number of other context files **902** in an arbitrary graph manner, as best determined by the content provider. For example, the consumer user context file **902***e* references two other context files, the "Looking for a Camera" context files **902***h*, and the "Shopping for a Camera" context file **902***i*. These context files more precisely focus on serving the user's intention, the former focusing on the information needs when a user is still looking for a camera and in need

of information to evaluate potential products. The latter context is appropriate when a particular camera has been selected and the user is now shopping for the camera based on price, availability, and other factors. Again, each of these context files **902** references different and more selective contexts. Thus, the "Looking for a Camera" context file **902***h* references a group of context files **902***k* pertaining to various types of reviews of digital cameras. The "Shopping for a Camera" context file **902***i* references context files **902***m*, **902***l* for comparing prices, and for comparing vendors. The context files **902** can also be arranged hierarchically through a series of directories.

[**0138**] As previously discussed, a context file may include query revision rules, and search engine control information that enables the context processor to programmatically tailor the user's query to the information needed, as indicated by the context. For example, once the user enters the "Looking for a Camera" context, that context file **902***h* may contain search control data that selects specific websites that contain consumer oriented camera reviews, as deemed appropriate by the vertical content provider. This control data would thus be used by the search engine system to select one or more document collections for targeting the query (or revised queries) thereto.

[**0139**] Similarly, the "Shopping for a Camera" context file **902***i* would include search control data that selects various price comparison engines to obtain current market prices on a given camera. These examples illustrate how selection of a context can programmatically vary the search query and search control data and parameters in order to better suit the user's information needs.

[**0140**] It is important to further point out here that the specific editorial decisions reflected in each context file **902**—how to revise a query based on whether the user is a professional or a consumer, or which sites to search depending on whether the context is shopping or looking—are made by each vertical content provider individually. This gives each vertical content provider—such as those with expertise in a particular field, such as digital cameras—the ability to define the contexts as they see fit, thereby using their own judgment, expertise, knowledge, and opinions to make the various determinations. Each vertical content provider can define very detailed and precisely crafted contexts, each of which can specifically control the operations of the programmable search engine in responding to a search query. Users ultimately benefit from this individuated capability because the vertical content providers to create a dynamic information "market": a market not merely for content itself, but for perspective, experience, and knowledge. That is, vertical content providers now offer users the ability to "search the world" through their own point of view, as suggested in FIG. **1** by the text "Search the web with digitalslr.org."

[**0141**] One mechanism for encapsulating the expertise and judgment of each vertical content provider is, at least in part, the site/page annotation file **900**. This context file **900** includes information variously categorizing or describing characteristics of sites or pages on the Internet. In addition to annotating a site or a page, a developer can also annotate all the pages that share a certain URL prefix, whether or not there is an actual page with that prefix. Each entry in the site/page annotation file **900** provides an identifier of a site or page, e.g., a URL, along with a number of tags or token identifying attributes, characteristics, weightings, or other qualitative or quantitative values. The tags can be explicitly typed (e.g., as <tag, value> pairs), or implicitly typed based on order and data format. A URL can specify a site or page completely, or in part as a URL prefix, for some portion of a web site. Such an annotation file can be provided using existing standard formats such as RSS (RDF Site Summary or Really Simple Syndication).

[**0142**] The following are some examples of the contents of a site/page annotation file. These examples might be provided, for example, via an RSS feed or by some other mechanism.

[**0143**] url, http://www.dealtime.com/xPR-Nikon_D100~RD-81887137412, descriptor, Review/NegativeReview, rank, 6, comment, Professional Photographer lists various shortcoming and compatibility problems

[**0144**] url, http://www.dealtime.com/xPR-Nikon_D100~RD-81887137412, descriptor, Review/ProfessionalPhotographerReview, rank, 0, comment, Professional Photographer is less thrilled than many others about the D100

[**0145**] url, http://www.dpreview.com/reviews/read_opinion_text .asp?prodkey=nikon_d100&opinion=15851, descriptor, Action, rank, 0, comment, Short review on using the D100 for sports photography

[**0146**] url, http://nikonimaging.com/global/news/, descriptor, News, rank, 3, comment, Nikon's web site. Lots of info, but hard to navigate

[**0147**] url, http://www.kenrockwell.com/tech/2dig.htm, descriptor, Guide, rank, 0, comment, Explains Digital SLRs vs Point and Shoots

[**0148**] url, http://www.luminous-landscape.com/tutorials/nikon-sn.shtml, descriptor, Review/ProfessionalPhotographerReview, rank, 8, comment, Extremely detailed, very technical, comparative review

[**0149**] url, http://www.photographyreview.com/, descriptor, Review, rank, 6, comment, Good all around site for photography buffs

[**0150**] url, gallery.photographyreview.com/showphoto, descriptor, Photos, rank, 8, comment, Good showcase of great photography with a wide range of cameras

[**0151**] url, http://www.olympusamerica.com/, descriptor, Manufacturer, rank, 10, comment, Olympus's web site. Well organized and informative

[**0152**] In this embodiment of a site/page annotation file **900**, each entry is a set of <name, value> pairs, as follows:

[**0153**] URL: provides the network address for where the site or page is located. Note that both specific pages within sites can be identified, as well as home pages for large sites.

[**0154**] Descriptor: a semantic label describing the site or page. The content provider is free to use any labels he or she chooses, since the query processing and post-processing operations are written in terms of rules that can operate on these same descriptors. In the above example, the vertical content provider has labeled various sites/pages to their content type (e.g. "Negative review" or "News" or "Pho-

tos"), as well as to the type of entity which provides the information (e.g., "Manufacturer"). Again, these descriptors are merely illustrative, and the selection of which particular descriptors are used to describe a site will be dependent in at least in part on the particular category or topic for the subject matter of the domain.

[0155] Referring back then first entry here is for a specifically identified page on a remote site (dealtime.com) that contains a "negative review" of the Nikon D100 camera.

[0156] The pre-processing and post-processing operations can use the tags as conditions for evaluation. For example, a post-processing rule in the "Negative Reviews" context file **902***n* would select for inclusion in the search results that had a tag "Negative Review/NegativeReview". The various tags shown above—Manufacturer, Guide, Photos, etc.—are merely illustrative of the scope and variety that can be used. The ability to tag any site or page with a semantic label allows for very powerful pre-processing and post-processing operations by the context processor.

[0157] In one embodiment, there is provided a common ontology of tags which can be used, either exclusively or in conjunction with a set of private tags defined by vertical content provider. The ontology includes a hierarchy of categories of information and content on Internet. One useful ontology is provided by the Open Directory Project, found at dmoz.org. All or a portion of such an ontology can be used for the tags. The ontology can be public, as in the OPD, or proprietary, or a combination of both.

[0158] Rank: Each entry can have a rank (or "score", "weight", etc.) a figure of merit as to the importance, quality, accuracy, usefulness, and the like of the particular page or site. This value is provided by the vertical content provider, again based on his or her own judgment and perspective. The rank value further allows the context processor to selectively include (or exclude) search results that have certain rank values, or to rank individual search results by this value as well.

[0159] Comment: Each entry can have a comment, explanation or description that the vertical content provider can use to further describe the page to the user. The comment allows the vertical content provider to further articulate the relationship between the page and the user's information need.

[0160] A given site or page can have multiple entries in the site/page annotation file **900**, each with its own descriptors, and other tags. For example, the first two entries above are for the same page, but with different descriptors, ranks, comments and so forth. When more than one entry matches a given URL, depending on the use, either both or the most specific entry is applied.

[0161] The URL, Descriptor, Rank, and Comment fields are illustrative of the types of information that can be included in the site/page annotation file **900**. The vertical content provider can define any number of other or additional attributes, and then define complementary pre-processing and post-processing rules that operate on such attributes. For example, other attributes that can be included in the site/page annotation file include:

[0162] Content Type: a designation of the type of site or page, such as guide, scientific article, government report, white paper, thesis, blog, and so forth.

[0163] Source Type: a designation of the source of the document, which maybe the same or different than the Tag. For example: government, commercial, non-profit, educational, personal, and so forth. An "Organization" attribute may serve a similar purpose.

[0164] Location: a designation of the country, state, country or other geographic region relevant to the page, using names, standard abbreviations, postal codes, geo-codes, or the like.

[0165] User Type: a designation of the intended type of user or audience for the site or page. For example, lay person, expert, homemaker, student, singles, married, elderly, and so forth.

[0166] The foregoing descriptors are themselves instances or specializations of a generic attribute type "tag". Accordingly, vertical content providers can choose to simply use the "tag" designation in association with a property value (e.g., tag, "Manufacturer"), or may use some specialization of tag, such as those listed above, or a combination of both approaches. This feature further enhances the flexibility and the extensibility of the present invention.

[0167] Any given page or site can have multiple different entries in the site/page annotation file. For example, the first two entries in the above list are for the same page, but have different tags, the first being a Negative Review, and the second being a Professional Photographer Review, different ranks, and different comments. This allows the vertical content provider to express the relevance of a give site for a particular context, rather than being limited to a single inclusion.

[0168] A second mechanism for capturing the knowledge and expertise of the vertical content provider is the knowledge base file **904**. The knowledge base file **904** is used to describe specific knowledge of concepts, facts, events, persons, and like. This information is encoded in a graph of object classes and instances thereof. A simple knowledge base file **904** could be as follows:

```
<KB>
<Class id="CameraModel"/>
<Class id="DigitalSLRCamera">
    <subClassOf ref="CameraModel"/>
</Class>
<DigitalSLRCamera id="NikonD100">
  <manufacturedIn ref="Japan"/>
  <name>D100</name>
  <name>Nikon D100</name>
  <manufacturer>Nikon</manufacturer>
  <brand>Nikon</brand>
  <format>SLR</format>
  <madein>Japan</madein>
  <modelyear>2003</modelyear>
  <megaPixels>6mp</megaPixels>
</DigitalSLRCamera>
<DigitalSLRCamera id="CanonDigitalRebel">
  <manufacturedIn ref="Japan"/>
  <name>EOS300D</name>
  <name>Digital Rebel</name>
  <manufacturer>Canon</manufacturer>
  <brand>Canon</brand>
  <format>SLR</format>
  <madein>Japan</madein>
  <modelyear>2003</modelyear>
  <megaPixels>6.5mp</megaPixels>
```

-continued

```
</DigitalSLRCamera>
</KB>
```

[0169] This knowledge base defines the class of "CameraModel", used to identify individual types of cameras. Each a each class had a class id, as shown. A class can then be a subclass of another class. Hence, the class "DigitalSLRCamera" is a subclass of the "CameraModel" class.

[0170] Instances of a class can then be defined as well. Here, two different instances of the class "DigitalSLRCamera" are defined by giving it a specific id, here "NikonD100" and "CanonDigitalRebel", and a listing of a variety of properties, such as their name, manufacturer, location of manufacture, model year, and so forth. The properties for each class are determined by the provider of the knowledge base file **904**, such as the vertical content provider.

[0171] The programmable search engine may maintain its own global knowledge base file as part of its global context files. This global knowledge base can provide an extensive database encapsulating a vast array of knowledge, concepts, facts, and so forth, as extracted from content on the Internet, provided by experts or editors, or any taken from existing databases. Vertical content providers can then make use of this global knowledge base by providing preprocessing and post-processing operations that make use of such knowledge base information, as further described below.

[0172] The context files **902** use a script or markup language to define the various pre-processing, search engine control, and post-processing operations. The various elements of the language are as follows:

Object Evaluation

[0173] The knowledge base file **904** can be used to evaluate whether particular objects have defined properties or attributes. In general, there are three basic types of objects that can be evaluated related to the knowledge base: queries, users, and search results. The form of the evaluation commands are generally the same.

[0174] The query evaluation commands for evaluating terms using the knowledge base file **904** are as follows:

```
<query.denot.property>property_value</query.denot.property>
<query.denot.InstanceOf>class_id</query.denot.InstanceOf>
<query>query_term</query>
```

[0175] The first type of term based evaluation is used to evaluate whether the concept expressed by one or more query terms matches some object in the knowledge base file that has the specified property with the specified property value. The context processor processes this command by traversing the knowledge base file **904** (as a graph, for example) until it finds an object having a property with the matching property value. For example, assume the knowledge base file **904** portion described above, and the query evaluation command:

[0176] <query.denot.Manufacturer>Nikon</query.denot-.Manufacturer>

[0177] and the input search query "D100".

[0178] Here, the query term "D100" matches the name of a camera instance in the knowledge base file **904**. The context processor than checks whether the Manufacturer property of that instance is "Nikon". Since it is, the query "D100" is said to denote a camera manufactured by Nikon, even if that is not specifically disclosed in the query term itself. Accordingly the query evaluation command is satisfied, and the context processor would then take an appropriate action that was dependent on this evaluation. As will be further illustrated below, a variety of different commands to the context processor can be made conditional based on the evaluation of the query evaluation command.

[0179] The second type of query evaluation command is query.denot.Instanceof. This command is evaluated to determine whether a particular query indicates that an instance of a class has been described in the query, rather than property. For example, consider the query evaluation command:

[0180] <query.denot.Instanceof>DigitalSLRCamera</query.denot>

[0181] where the user query is "8mpSLR".

[0182] Here, the query is decomposed into terms "8mp" and "SLR", and these are checked against the property values for the objects in the knowledge base file. In this example, these properties match the properties for the Nikon D100 camera, satisfying the query evaluation command. Again, the context processor would undertake whatever command was conditioned on the evaluation command.

[0183] The last type of query evaluation command <query>query_term</query> is the simplest. The query evaluation command is satisfied if an input search query term matches the query_term.

[0184] As noted above, the context files may used with any combination of query evaluation commands as conditional triggers for further context processing. Example of these will be further described below. As with the evaluation of queries, so too can users and search results be evaluated for their properties, with respect to defined any defined class in the knowledge base file. Thus, the attributes of user can be evaluated with the following command

[0185] <user.property>property value</user.property>

[0186] where property refers to any available property of the user, such as user name, login, account number, location, IP address, site activity and history. (e.g., clicks, focus, page dwell time) and so forth. Some of these properties can be locally available from the knowledge base file **904**. Alternatively, the property information can be extracted (e.g., queried) from any accessible legacy database (e.g., a customer database, account database, registration database, or other data source), which exports an appropriate programmatic interface. Other properties, such as site activity, are made available from site tracking tools that monitor each user's activity at the vertical content site.

[0187] Users can also be evaluated for membership in classes, using the following:

[0188] <user.InstanceOf>class_id</user.instanceOf>

Here, a class of users (e.g., "Professional") can be defined in the knowledge base file **904**, and the properties of the

current user compared by the context processor against the properties of an identified class for match in values. If a property match is found, the user is deemed a member of the class.

[0189] Similarly, any search result can be evaluated as well, as to its properties, as defined in either the source/page annotation file **900** (or alternatively, in its metatags). Here, the evaluation command would take the form:

```
<result.tag>tag_value</result.tag>
<result.tag.InstanceOf>class_id</result.tag.InstanceOf>
```

[0190] As a default <result.tag> may be abbreviated to <tag>.

[0191] In the first command, a given search result (or set thereof) can be evaluated with respect to its properties, such as content type, date, source, user type, etc. This outcome of the evaluation can be used to control further context processing. Similarly, search results can be evaluated using the second command syntax to determine if they are instances of various classes defined in the knowledge base file **904**.

[0192] These following context processing operations can be executed unconditionally, or conditionally based on any of the foregoing types of evaluation operations (e.g., evaluations of query terms, users, or search results).

Query Modification

[0193] There are two basic types of query modification rules, those that augment or add terms to a query, and those that replace query terms. The following is example syntax for the query modifier command:

```
<QueryModifier type="augment" value="query term"/>
<QueryModifier type="replace" query="query term"
value="replacement term"/>
```

[0194] The type attribute defines either an augmentation or replacement type query modification. The value attribute includes the query term that is to be added to the user's original input search query, or that is to replace the input search query. The query attribute is optional. If present, then the context processor scans the search query and replaces the any term matching the query term with the replacement term. This is useful, for example, to correct misspellings, expand abbreviations (or contrawise use abbreviations in place of terms), and other in place adjustments. If the query attribute is missing, then the entry query string is replaced by the replacement term. Of course, the replacement term can include any number of terms.

[0195] Query modification can made conditional on any of the evaluation commands. For example:

```
<QueryModifier type="augment" value="Digital SLR">
<query.denot.InstanceOf>DigitalSLRCamera</query.denot>
    </QueryModifer>
```

This example would reformulate a query, say the query "D100" to include another query "Digital SLR" since the term "D100" denotes an instance of a digital SLR camera, according to the knowledge base file **904**.

[0196] As another example:

```
<QueryModifier type="augment" value="Professional reviews">
<user.property>professional</user.property>
</QueryModifer>
```

In this example, assume again the user's query is "D100." Here, the properties of the current user are evaluated. If the user is determined to be "professional", based on properties available from the browser, site activity history, login and password, etc. For example, if the user access a number of pages in the vertical content site dedicated to professional or expert level information (e.g., detailed technical pages), then the user may be inferred to be a "professional" user, even though no other information is known about the user's identity. In this case, the query is reformulated to include the term "professional reviews" even though the user did not include these terms in the query.

[0197] These are but a few examples of a how a vertical content provider can extend and improve the user's queries based on his own expertise and the flexible context processing operations.

References to Related Contexts

A context file **902** can reference or include another context file **902**, as described above, to form an arbitrary graph of connections. Several elements are used for referencing context files.

A context file can include another context file, as follows:

[0198] <include scr="path name">

[0199] The include command references another context file **902** as being included in the current context file. The context processor will read the included context file and process all of the instructions therein. Pathname identifies the location of included context file **902**. Included context files **902** can be used for any type of context processing operation.

[0200] A context file can also identify a related context file, as follows:

```
<relContext href="path name">
   <anchorText>context description<anchorText>
</relContext>
and
<relContext href="path name">context description
</relContext>
```

[0201] The relContext command identifies a related context for the current context file. The relContext command can be used in both pre-processing and post-processing operations. Examples of the use of related contexts in postprocessing operations are illustrated in FIG. **10**, and in FIGS. **2** and **3**. The context description is anchor text that the user will see in the browser. When selected, the identified

related context file is retrieved and processed. The first type of related context command is used to define related contexts for varying types of information needs. FIG. **2** illustrates this type of related context via related context links **204**. The first link **204** there is associated with a related context file **902** (e.g., context file **902***h*) that includes the following instructions:

```
<relContext href=" /chooseCamera">
    <anchorText>If you are trying to decide which camera
to buy ...</anchorText>
</relContext>
```

This command is processed by the context processor when the link **204** on the anchor text is selected, and the corresponding context file "cameras/chooseCamera" is retrieved and processed. The resulting page is illustrated in FIG. **3**.

[**0202**] The relContext command may also be used with the various types of evaluation commands, to make the reference to the related context conditional. For example:

```
<relContext href=" /chooseCamera">
    <query.denot.instanceOf>DigitalSLRCamera</query.denot
    .instanceOf>
    <anchorText>If you are trying to decide which camera
to buy ...</anchorText>
</relContext>
```

[**0203**] Here, the related context DigitalSLRCamera is accessed here only if the query.denote command evaluates true, that is where the query terms denote an instance of a model of digital camera listed in the knowledge base file **904**. Similar conditional evaluations can be based on the properties of the user or the properties of the search results.

[**0204**] The second type of related context command is used to define related contexts that appear as annotations in conjunction with search results. This type of related context is illustrated in FIG. **2** by related context links **206**. For example, the related context file **902***h* that generated FIG. **2** also includes the following instructions:

```
<relContext href="cameras/Manufacturer">More Manufacturer
Pages</relContext>
```

Here, the anchor text "More Manufacturer Pages" is then linked to the associated context file **902**, which contains further instructions to searching and displaying pages for digital camera manufacturers.

The relContext command takes as an href any valid URL, and thus, can also reference any available Internet site. For example, the relContext command can directly link to an online encyclopedia or dictionary to provide an annotation for a search result that would provide a detailed explanation of the result.

[**0205**] In pre-processing operations, a second type of cross reference to related context is used, context redirection. The command format for the context redirection command is as follows:

```
    <contextRedirect href="pathname">redirection condition
    *</contextRedirect>
```

[**0206**] Again, pathname indicates the location of another context file to be processed if certain redirection conditions are met. The redirection conditions (one or more as indicated by "*") can be based on any available information about the query (e.g., query terms, or information dependent thereon), the user (e.g., IP address, login, site click through history, prior purchases), or other programmatically available information.

[**0207**] In one embodiment the redirection conditions can be based on the any evaluation commands previously discussed:

```
    <query.denot.property>property_value</query.denot.property>
    <query.denot.InstanceOf>class_id</query.denot.InstanceOf>
    <query>query_term</query>
<user.property>property_value</user.property>
<user.InstanceOf>class_id</user.instanceOf>
<result.tag>tag_value</result.tag>
<result.tag.InstanceOf>class_id</result.tag.InstanceOf>
```

[**0208**] For example, assume the knowledge base file **904** portion described above. Further, assume the redirection command:

```
<contextRedirect href="Nikon_cameras">
    <query.denot.Manufacturer>Nikon</query.denot.Manufacturer>
</contextRedirect>
```

and the input search query "D100".

As above, the query evaluation command is positively evaluated, since the query term "D100" matches the name of a camera instance in the knowledge base file **904**, which instance has the Manufacturer property value "Nikon". The context processor thus executes the context redirection command and accesses the context file "Nikon_cameras" for further processing. This capability allows the vertical content provider to his or her own knowledge base to analyze queries and reformulate them on behalf of the user.

[**0209**] The user evaluation user.InstanceOf can likewise be used to redirect context processing based on the particular user properties For example, consider the redirection command:

```
    <contextRedirect href="NegativeProfessionalReviews">
        <user.InstanceOf>Professional_User</user.InstanceOf>
    </contextRedirect>
```

[**0210**] Here, the properties of the user can be ascertained from the knowledge base file **904**, and other information as

described (e.g., site history). If the user is determined to be a professional user, then the context processor accesses and processes the NegativeProfessionalReviews context file.

[0211] As mentioned, any number of redirection conditions (e.g. evaluations) can be used together in a context redirection command such as:

```
<contextRedirect href="Recommended_SLR_cameras">
    <query.denot.megapixels
matchType="greaterThanOrEqualTo">6mp</query.denot.megapixels>
    <query.denot.megapixels
matchType="lessThanOrEqualTo">8mp</query.denot.megapixels>
    <query.denot.modelyear>2005</query.denot.modelyear>
</contextRedirect>
```

which would effect the context redirection only when all of the redirection conditions are satisfied, e.g., for a query containing the terms which denote digital SLR cameras with between 6mp and 8mp, for the 2005 model year.

[0212] The context redirection is particularly powerful when combined with the query modification rules, previously discussed. A vertical content provider can define a number of context redirections based on query terms, each of redirects the context processor to an appropriate context file, depending on say, whether the query denotes shopping for a camera versus seeking customer warranty information. In the respective target context files, specific query modification rules would then be processed to reformulate the query as most appropriate given the identified context.

Restriction

[0213] In post-processing operations, the context files can be used to control the scope, number, or types of results and entries that are provided to the user. To this end, the context files can include conditional instructions that define various types of restrictions (e.g., filters). These restrictions are provided by the restriction command. This command has the following syntax:

```
<Restriction count="n">
    restriction condition*
    restriction action*
</Restriction>
```

[0214] The restriction condition operates in a similar manner to the redirection condition previously discussed. Here, the restriction condition is evaluated with respect to the attributes (tags), if any, associated with the search results, as compared to the entries in the site/page annotation file. Any attribute (or set of attributes) can be used as restriction conditions, such as the type, source, year, location, of a document or page, to name but a few. The context processor receives the search results (here a set of candidate search results) from the search engine, and compares each candidate result (be it a site, page, media page, document, etc.) with the entries listed in the site/page annotation file **900**. Only those candidate results which are listed in the annotation file **904** and have the specified matching attributes are included in the context-processed search results. The restriction count is an optional parameter and indicates how many of the matching results are to be included in the context-processed search results. If left out, then all matching results are included.

[0215] The restriction action is an optional parameter that specifies a further action to take if the restriction condition is met. This action includes, for example, annotating the search results with a link to a related context (using the relContext command), such as links B**06** illustrated in FIG. **2**.

[0216] Consider the following example:

```
<Restriction count="2">
    <descriptor>Review</descriptor>
    <rank>5+</rank>
    <relContext href="Reviews">More Review</relContext>
</Restriction>
<Restriction count="2">
    <descriptor>Guide</descriptor>
    <rank>5+</rank>
    <relContext href="Guides">More Guides</relContext>
</Restriction>
```

[0217] Assume that the search query was a general query on "digital cameras", and that the search results returned 1,000,000 pages covering everything from manufacturers and retailers of digital cameras, to online user forums and services for printing photographs. Since the user's search was quite general, the vertical content provider can use the post-processing to provide a selection of a number of different types of search results, as illustrated, for example in FIG. **2**. In processing the above code example then, the first restriction command causes the context processor to select the first two search results that have matching entries (i.e., matching URLs or portions thereof) in the site/page annotation file **900** and include the descriptor "Review". The context processor also uses the restriction action for the related context, to annotate these two search results with a link to related context file "Reviews", with the link labeled "More reviews."FIG. **2** shows an example of such annotation link B**06**.

[0218] The second restriction causes the context processor to select the first two search results that have matching entries in the site/page annotation file and include the descriptor "Guide." The context processor would then use the restriction action to annotate these results with a link to the related context file "Guides."

[0219] As mentioned previously, the context processing operations can undertaken by multiple different entities in the system, including at the client device, the vertical content site, and the programmable search engine, each using their own locally available context files. Thus, all of the above describe features can be effectively integrated within and between different system entities. For example, a vertical context provider can define a context file that defines various context redirections using the redirection condition based on the global knowledge base files. This enables the vertical content provider to leverage the global knowledge base, but add their own personal perspective and judgment to its underlying facts.

Search Engine Control Data

[0220] Finally, context files **902** can contain instructions that control the operation of the programmable search

engine itself in terms the selection of which particular document collections to be searched, and various algorithmic or parametric settings for the search engine. Selection of a document collection for searching is provided by the following command:

```
<Corpus ref="document_collection">
    //other context operations//
</Corpus>
```

The corpus command takes as its argument a reference to the name (or URL) or a selected document collection. The document collection name is mapped (either locally, or by the programmable search engine) to document collection and corresponding index available to the programmable search engine (e.g. particular index in the content server/index **870**).

The corpus command can be made conditional using any of the foregoing described evaluation commands, as well as including any of the restriction, redirectioh, related context, and so forth.

[0221] For example, a particular document collection may be selected where the query is determined using the evaluation commands to include certain keywords or instances of objects in the knowledge base. Thus, a query that is evaluated to include a query term denoting a scientific term, like "Heloderma suspectum", or a medical term, would then cause a selection of an appropriate scientific literature database.

[0222] Control of search engine parameters is via the SearchControlParams operations. In general, most modern search engines use a number of different attributes of a search query and the individual indexed documents (e.g., frequencies of terms in URL, anchor text, body, page rank etc.) to determine which documents best satisfy the query. The documents are then ranked accordingly. A ranking function is essentially a weighted combination of the various attributes. Normally, the weights of the attributes are fixed, or at least not externally controllable by third parties. The SearchControlParam however gives vertical content providers access to these weights. The syntax is as follows:

```
<SearchControlParams>
    <attribute-name>weight</attribute-name>
    <attribute-name>weight</attribute-name>
    ...
</SearchControlParams>
```

[0223] Here, attribute-name is the name of the particular attribute used by the search engine to calculate a relevance ranking. The specific attribute names are disclosed by the programmable search engine provider, since they are internal to that provider's own engine. Typical attributes, as indicated above including term frequency in URL, term frequency in body, term frequency in anchor text, term frequency in markup, page rank. The SearchControlParams operator can work with any exposed attribute or parametric control of a programmable search engine, and thus the foregoing are understood to be merely exemplary. The weights used in this operator can be either normalized or non-normalized, and in the latter case, the input weights can be internally normalized by the context processor or by the search engine itself. A vertical content provider need not specify weights for all the attributes the search engine uses, but only those of interest to the provider of the context file.

[0224] The present invention has been described in particular detail with respect to one possible embodiment. Those of skill in the art will appreciate that the invention may be practiced in other embodiments. First, the particular naming of the components, capitalization of terms, the attributes, data structures, or any other programming or structural aspect is not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, formats, or protocols. Further, the system may be implemented via a combination of hardware and software, as described, or entirely in hardware elements. Also, the particular division of functionality between the various system components described herein is merely exemplary, and not mandatory; functions performed by a single system component may instead be performed by multiple components, and functions performed by multiple components may instead be performed by a single component.

[0225] Some portions of above description present the features of the present invention in terms of algorithms and symbolic representations of operations on information. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. These operations, while described functionally or logically, are understood to be implemented by computer programs. Furthermore, it has also proven convenient at times, to refer to these arrangements of operations as modules or by functional names, without loss of generality.

[0226] Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as "calculating" or "determining" or "identifying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0227] Certain aspects of the present invention have been described using commands, mnemonics, tokens, formats, syntax, and other programming conventions. The particular selections of the names, formats, syntax, and like are merely illustrative, and not limiting. Those of skill in the art can readily construct alterative names, formats, syntax rules, and so forth for defining context files and programming the operations a programmable search engine via context processing.

[0228] Certain aspects of the present invention include process steps and instructions described herein in the form of an algorithm. It should be noted that the process steps and instructions of the present invention could be embodied in software, firmware or hardware, and when embodied in software, could be downloaded to reside on and be operated from different platforms used by real time network operating systems.

[0229] The present invention also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored on a computer readable medium that can be accessed by the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magneticoptical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0230] The algorithms and operations presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may also be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will be apparent to those of skill in the art, along with equivalent variations. In addition, the present invention is not described with reference to any particular programming language. It is appreciated that a variety of programming languages may be used to implement the teachings of the present invention as described herein, and any references to specific languages are provided for disclosure of enablement and best mode of the present invention.

[0231] Finally, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

What is claimed is:

1. A method of processing a search query for a search engine to provide a set of search results to the search query, the method comprising:

receiving a search query for a user from a host system;

retrieving, from a repository, at least one previously stored context associated with at least one of the user and a host system;

processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

providing the processed search results to the user.

2. The method of claim 1, wherein processing the query comprises at least one of:

pre-processing the query using the identified context by programmatically altering execution of the search engine on the query, to produce search results responsive to the pre-processed query; and

post-processing the search results using the identified context by programmatically altering the search results.

3. The method of claim 1, wherein processing the query comprises:

pre-processing the query using the identified context by programmatically altering execution of the search engine on the query, to produce search results responsive to the pre-processed query; and

post-processing the search results using the identified context by programmatically altering the search results.

4. The method of claim 1, wherein retrieving the previously stored context is performed by a first context server, and wherein the previously stored context was stored by an entity other than the first context server.

5. A method of processing a search query for a search engine to provide a set of search results for the search query, the method comprising:

receiving a search query for a user from a host system;

receiving at least one context associated with at least one of the user and a host system;

storing the received context;

processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

providing the processed search results to the user.

6. The method of claim 5, wherein storing the received context comprises storing the received context in a repository accessible to a plurality of context servers.

7. The method of claim 5, wherein storing the received context comprises storing the received context in a database.

8. A method of processing a search query for a search engine to provide a set of search results for the search query, the method comprising:

receiving a search query for a user from a host system;

receiving at least one context associated with at least one of the user and a host system;

determining whether the received context corresponds to a persistent characteristic of at least one of the user and a host system;

responsive to the received context corresponding to a persistent characteristic of at least one of the user and a host system, storing the received context;

processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

providing the processed search results to the user.

9. The method of claim 8, wherein storing the received context comprises storing the received context in a repository accessible to a plurality of context servers.

10. The method of claim 8, wherein storing the received context comprises storing the received context in a database.

11. A method of sharing contexts among context servers, the method comprising:

receiving, at a first entity, at least one context associated with at least one of a user and a host system; and

sharing the received at least one context with at least a second entity different from the first entity, wherein the second entity comprises a context server for processing the query using the context.

12. The method of claim 11, wherein the first entity comprises a context server.

13. The method of claim 11, wherein sharing the context is performed responsive to determining that the received context corresponds to a persistent characteristic of at least one of a user and a host system.

14. The method of claim 11, wherein sharing the context comprises storing the context in a repository accessible to the second entity.

15. The method of claim 11, wherein sharing the context comprises storing the context in a database.

16. A computer program product for processing a search query for a search engine to provide a set of search results to the search query, the computer program product comprising:

a computer-readable medium; and

computer program code, encoded on the medium, for:

receiving a search query for a user from a host system;

retrieving, from a repository, at least one previously stored context associated with at least one of the user and a host system;

processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

providing the processed search results to the user.

17. The computer program product of claim 16, wherein processing the query comprises at least one of:

pre-processing the query using the identified context by programmatically altering execution of the search engine on the query, to produce search results responsive to the pre-processed query; and

post-processing the search results using the identified context by programmatically altering the search results.

18. The computer program product of claim 16, wherein processing the query comprises:

pre-processing the query using the identified context by programmatically altering execution of the search engine on the query, to produce search results responsive to the pre-processed query; and

post-processing the search results using the identified context by programmatically altering the search results.

19. The computer program product of claim 16, wherein retrieving the previously stored context is performed by a first context server, and wherein the previously stored context was stored by an entity other than the first context server.

20. A computer program product for processing a search query for a search engine to provide a set of search results for the search query, the computer program product comprising:

a computer-readable medium; and

computer program code, encoded on the medium, for:

receiving a search query for a user from a host system;

receiving at least one context associated with at least one of the user and a host system;

storing the received context;

processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

providing the processed search results to the user.

21. The computer program product of claim 20, wherein storing the received context comprises storing the received context in a repository accessible to a plurality of context servers.

22. The computer program product of claim 20, wherein storing the received context comprises storing the received context in a database.

23. A computer program product for processing a search query for a search engine to provide a set of search results for the search query, the computer program product comprising:

a computer-readable medium; and

computer program code, encoded on the medium, for:

receiving a search query for a user from a host system;

receiving at least one context associated with at least one of the user and a host system;

determining whether the received context corresponds to a persistent characteristic of at least one of the user and a host system;

responsive to the received context corresponding to a persistent characteristic of at least one of the user and a host system, storing the received context;

processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

providing the processed search results to the user.

24. The computer program product of claim 23, wherein storing the received context comprises storing the received context in a repository accessible to a plurality of context servers.

25. The computer program product of claim 23, wherein storing the received context comprises storing the received context in a database.

26. A computer program product for sharing contexts among context servers, the computer program product comprising:

a computer-readable medium; and

computer program code, encoded on the medium, for:

receiving, at a first entity, at least one context associated with at least one of a user and a host system; and

sharing the received at least one context with at least a second entity different from the first entity, wherein the second entity comprises a context server for processing the query using the context.

27. The computer program product of claim 26, wherein the first entity comprises a context server.

28. The computer program product of claim 26, wherein sharing the context is performed responsive to determining that the context corresponds to a persistent characteristic of at least one of a user and a host system.

29. The computer program product of claim 26, wherein sharing the context comprises storing the received context in a repository accessible to the second entity.

30. The computer program product of claim 26, wherein sharing the received at least one context comprises storing the received at least one context in a database.

31. A system for processing a search query for a search engine to provide a set of search results to the search query, the system comprising:

an input mechanism, for receiving a search query for a user from a host system;

a shared context repository, for storing contexts;

a context server, for, responsive to the input mechanism receiving a search query, retrieving from the shared context repository at least one previously stored context associated with at least one of the user and a host system;

a context processor, coupled to the context server, for processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

an output mechanism, coupled to the context processor, for providing the processed search results to the user.

32. The system of claim 31, wherein the context processor performs at least one of:

pre-processing the query using the identified context by programmatically altering execution of the search engine on the query, to produce search results responsive to the pre-processed query; and

post-processing the search results using the identified context by programmatically altering the search results.

33. The system of claim 31, wherein the context processor performs at least one of:

pre-processing the query using the identified context by programmatically altering execution of the search engine on the query, to produce search results responsive to the pre-processed query; and

post-processing the search results using the identified context by programmatically altering the search results.

34. The system of claim 31, wherein the previously stored context was stored by an entity other than the context server.

35. A system for processing a search query for a search engine to provide a set of search results for the search query, the system comprising:

an input mechanism, for receiving a search query for a user from a host system;

a search engine interface, for identifying a context for the search query;

a shared context repository, for storing the identified context;

a context processor, for, responsive to the input mechanism receiving a search query, processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

an output mechanism, coupled to the context processor, for providing the processed search results to the user.

36. The system of claim 35, wherein the shared context repository is accessible to a plurality of context servers.

37. The system of claim 35, wherein the shared context repository comprises a database.

38. A system for processing a search query for a search engine to provide a set of search results for the search query, the system comprising:

an input mechanism, for receiving a search query for a user from a host system;

a search engine interface, for identifying a context associated with at least one of the user and a host system;

a shared context repository, for, responsive to the received context corresponding to a persistent characteristic of at least one of the user and a host system, storing the received context;

a context processor; for, responsive to the input mechanism receiving a search query, processing the query using the identified context by programmatically altering at least one of:

execution of the search engine on the query; and

the search results;

to produce processed search results; and

an output mechanism, coupled to the context processor, for providing the processed search results to the user.

39. The system of claim 38, wherein the shared context repository is accessible to a plurality of context servers.

40. The system of claim 38, wherein the shared context repository comprises a database.

41. A system for sharing contexts among context servers, the system comprising:

a search engine interface, associated with a first entity, for identifying at least one context associated with at least one of a user and a host system; and

a shared context repository, for storing the identified context and for providing it to a context server associated with a second entity different from the first entity.

42. The system of claim 41, wherein the shared context repository provides the identified context responsive to a determination that the context corresponds to a persistent characteristic of at least one of a user and a host system.

43. The system of claim 41, wherein the shared context repository comprises a database.

* * * * *