US 20050055341A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0055341 A1

Haahr et al. (43) Pub. Date: Mar. 10, 2005

(54) SYSTEM AND METHOD FOR PROVIDING SEARCH QUERY REFINEMENTS

(76) Inventors: **Paul Haahr**, San Francisco, CA (US); **Steven Baker**, Mountain View, CA (US)

Correspondence Address:
PATRICK J S INOUYE P S
810 3RD AVENUE
SUITE 258
SEATTLE, WA 98104 (US)

(21) Appl. No.: 10/668,721

(22) Filed: Sep. 22, 2003

Related U.S. Application Data

(60) Provisional application No. 60/500,539, filed on Sep. 5, 2003.

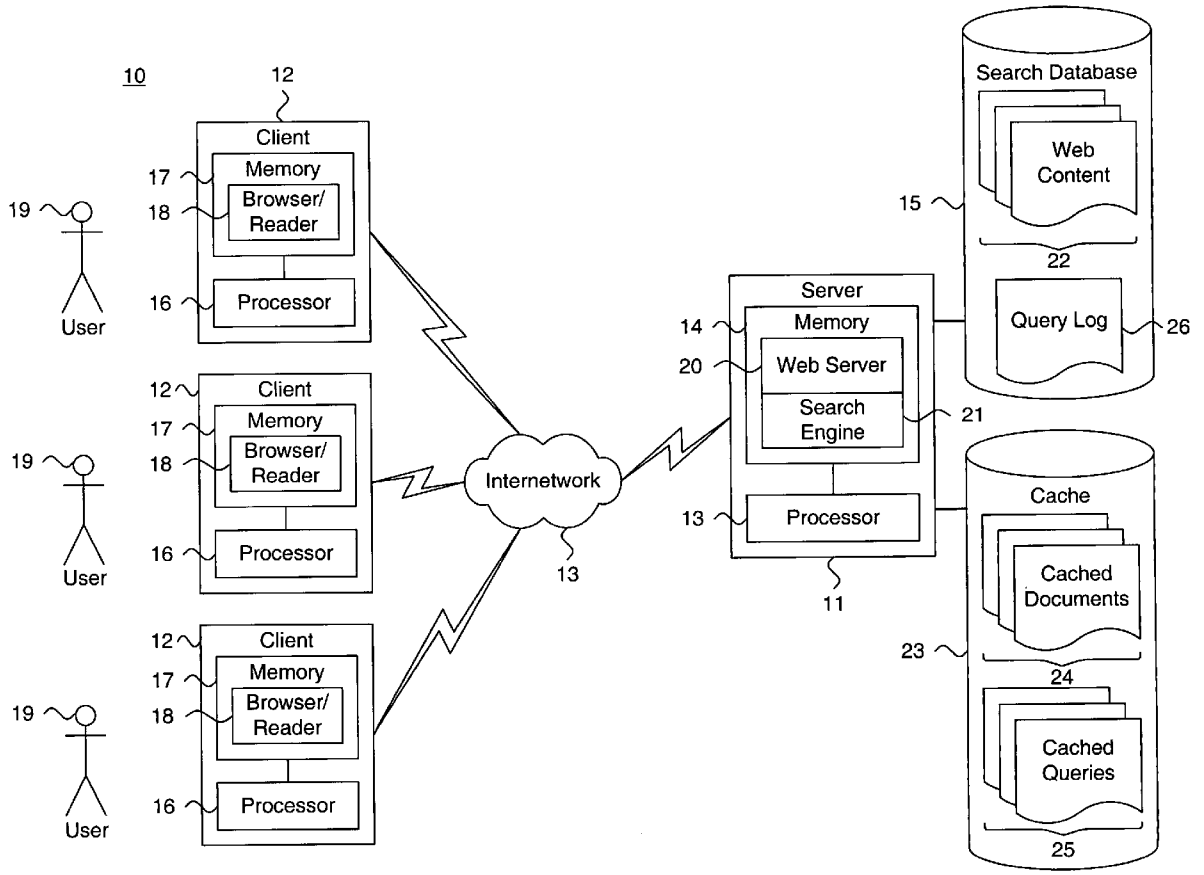
Publication Classification

(51) Int. Cl.[7] G06F 17/30

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

(57) **ABSTRACT**

A system and method for providing search query refinements are presented. A stored query and a stored document are associated as a logical pairing. A weight is assigned to the logical pairing. The search query is issued and a set of search documents is produced. At least one search document is matched to at least one stored document. The stored query and the assigned weight associated with the matching at least one stored document are retrieved. At least one cluster is formed based on the stored query and the assigned weight associated with the matching at least one stored document. The stored query associated with the matching at least one stored document are scored for the at least one cluster relative to at least one other cluster. At least one such scored search query is suggested as a set of query refinements.



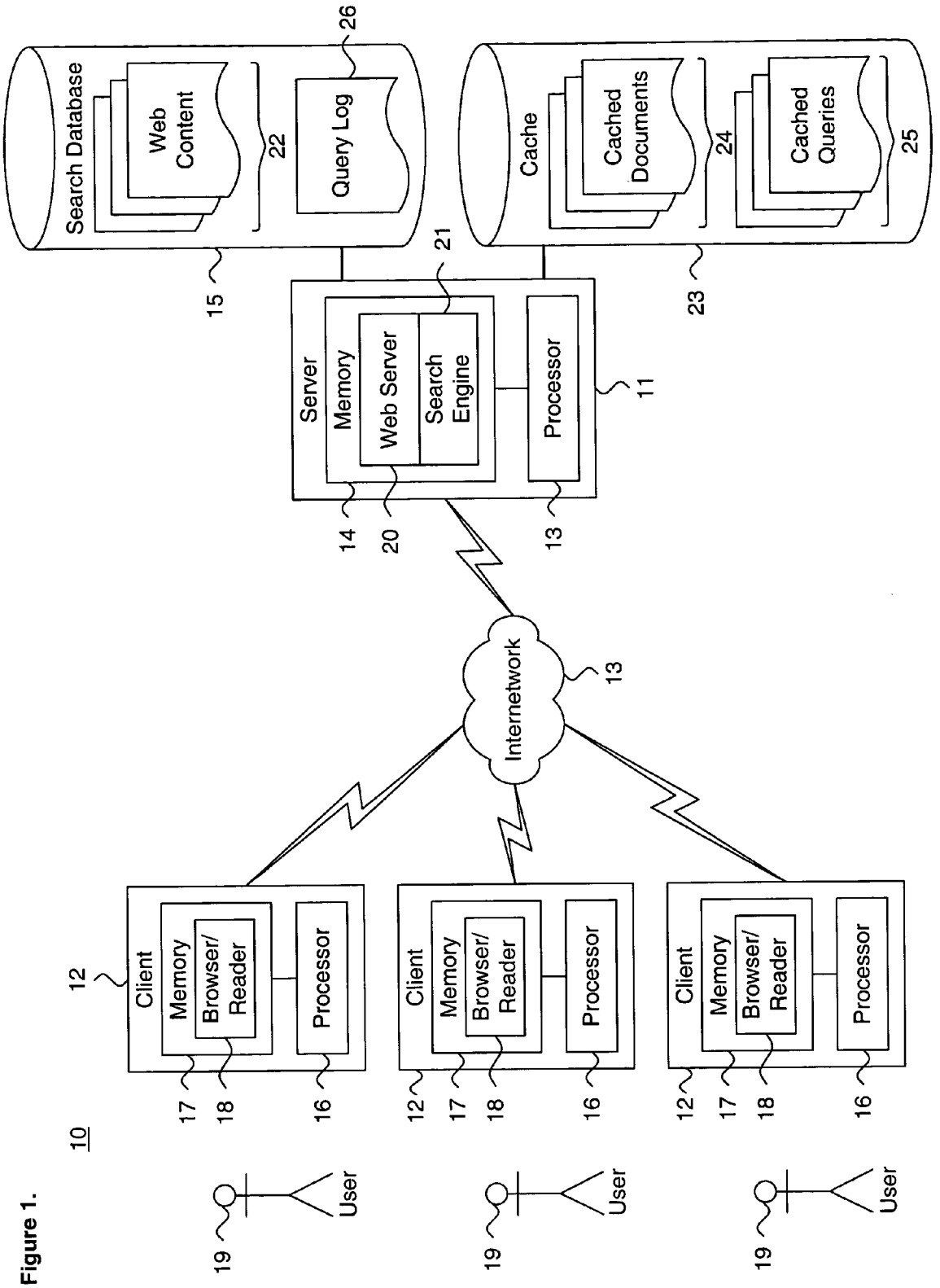

Search Database
Web Content
Query Log
22
26
15
Cache
Cached Documents
Cached Queries
24
25
23
21
Server
Memory
Web Server
Search Engine
Processor
11
14
20
13
Internetwork
13
10
12
Client
Memory
Browser/ Reader
Processor
17
18
16
User
19

Figure 1.

Figur 2.

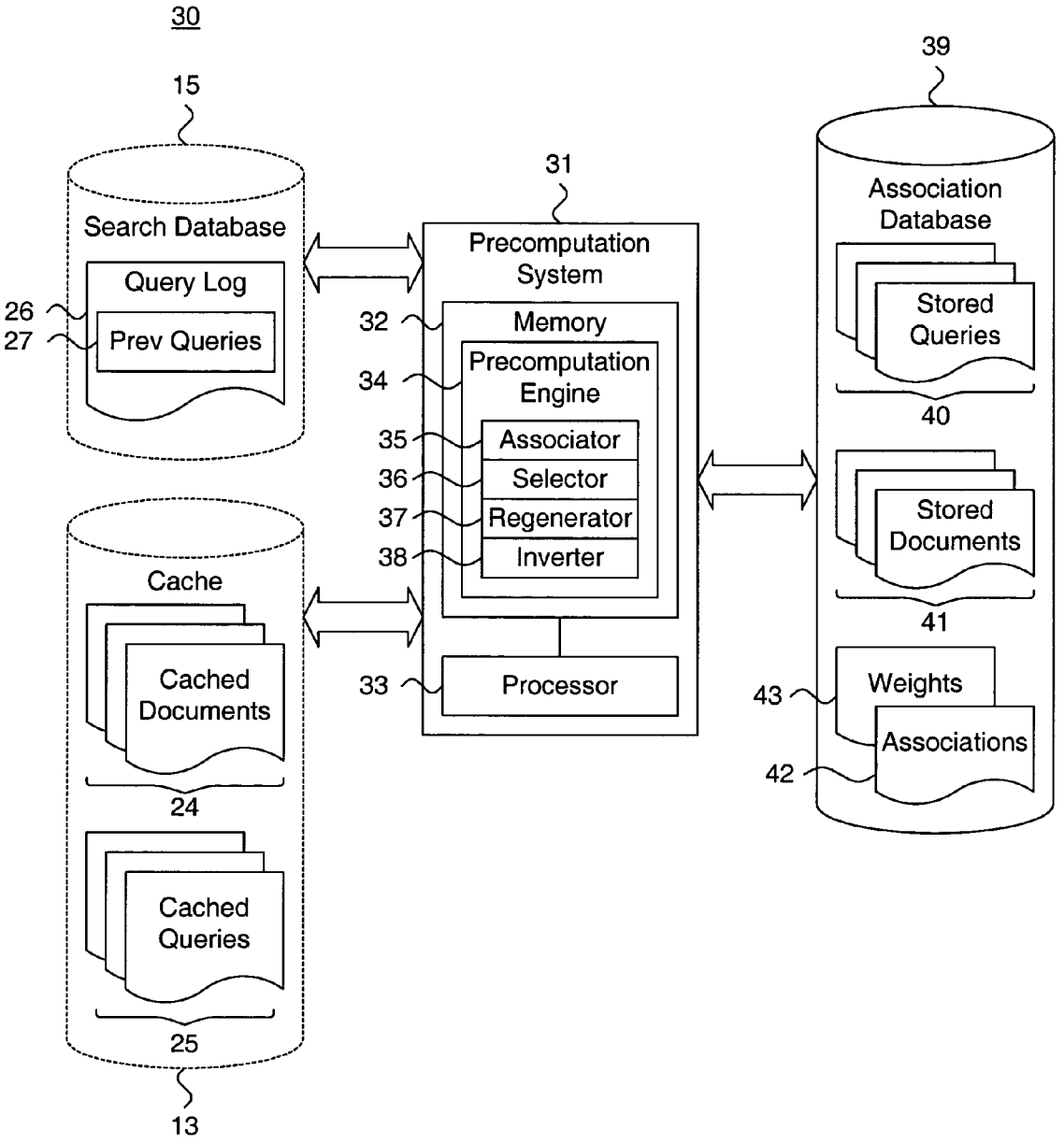
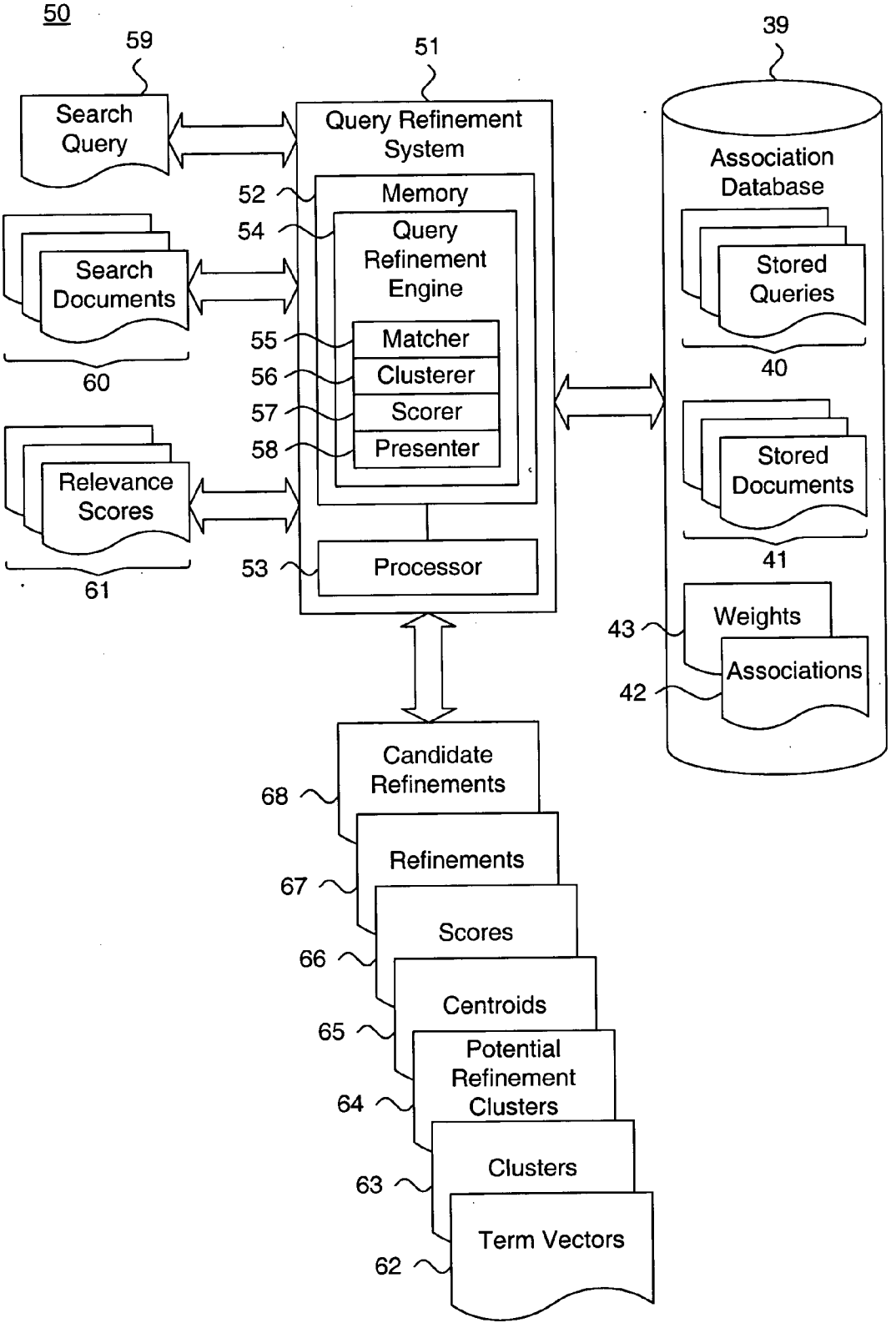

30
15
Search Database
Query Log
26
27
Prev Queries
Cache
Cached Documents
24
Cached Queries
25
13
31
Precomputation System
32
Memory
34
Precomputation Engine
35
Associator
36
Selector
37
Regenerator
38
Inverter
33
Processor
39
Association Database
Stored Queries
40
Stored Documents
41
43
Weights
Associations
42

Figure 3.


50
59
Search Query
51
Query Refinement System
39
Association Database
52
Memory
54
Query Refinement Engine
Search Documents
60
55
Matcher
56
Clusterer
57
Scorer
58
Presenter
Stored Queries
40
Relevance Scores
61
53
Processor
Stored Documents
41
43
Weights
42
Associations
Candidate Refinements
68
Refinements
67
Scores
66
Centroids
65
Potential Refinement Clusters
64
Clusters
63
Term Vectors
62

Figur 4.

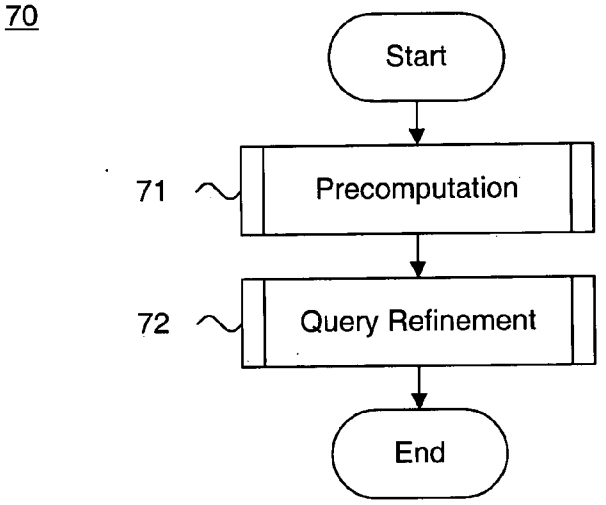
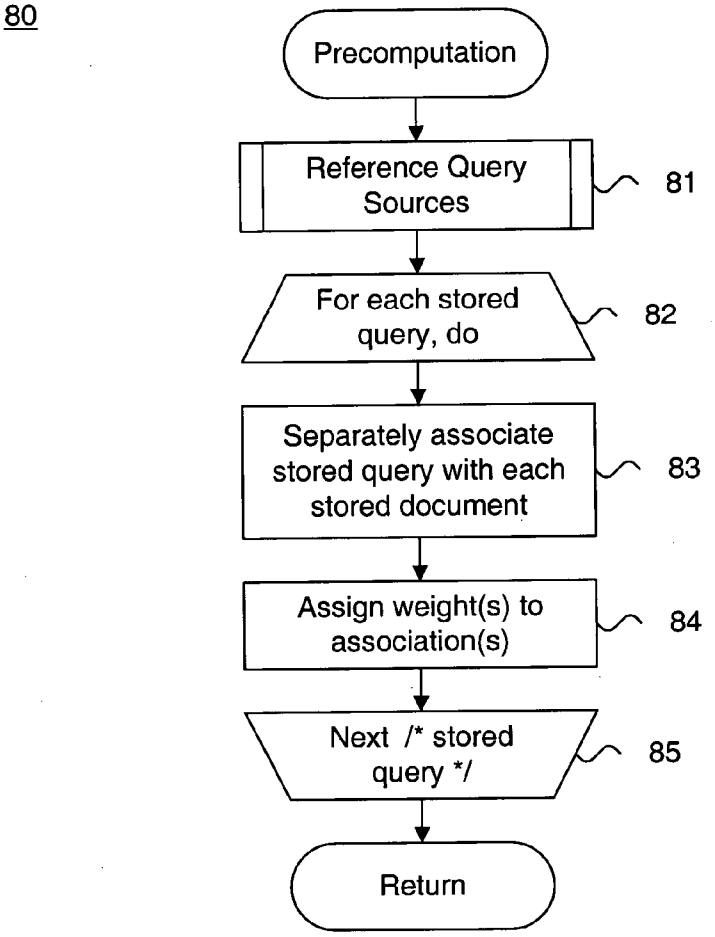


Figure 5.



Figur 6.

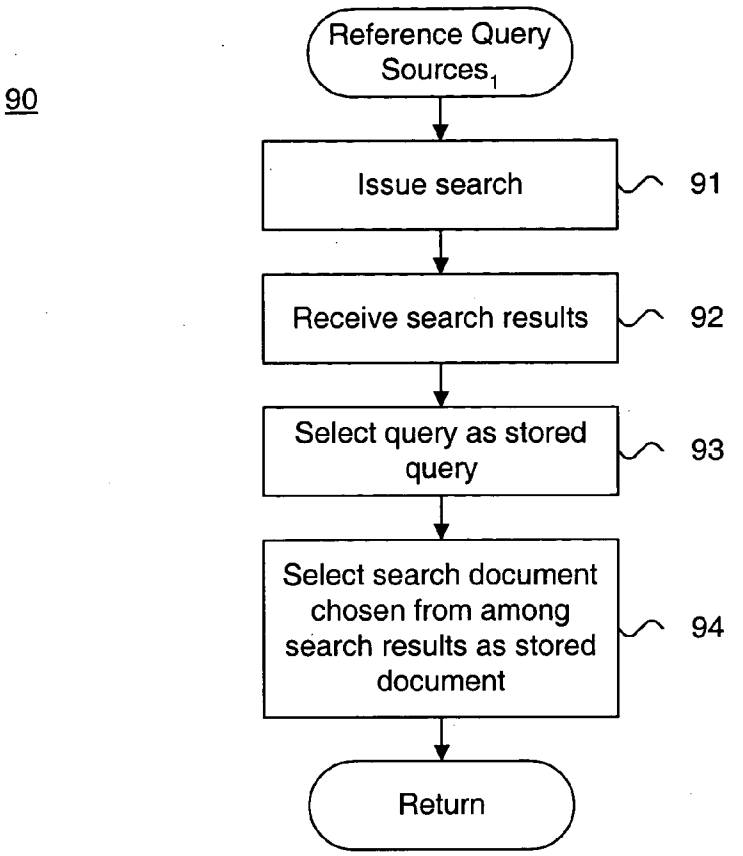
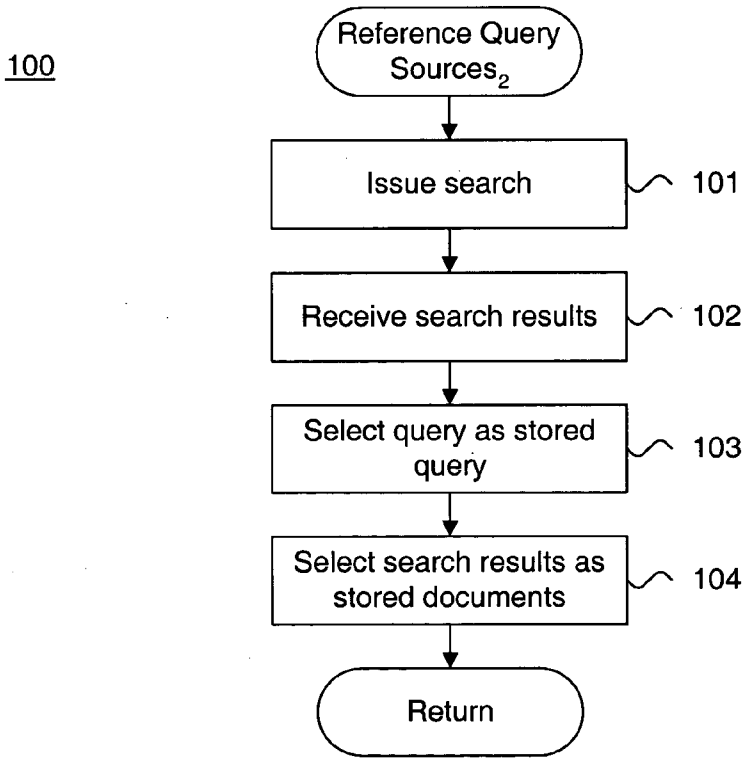
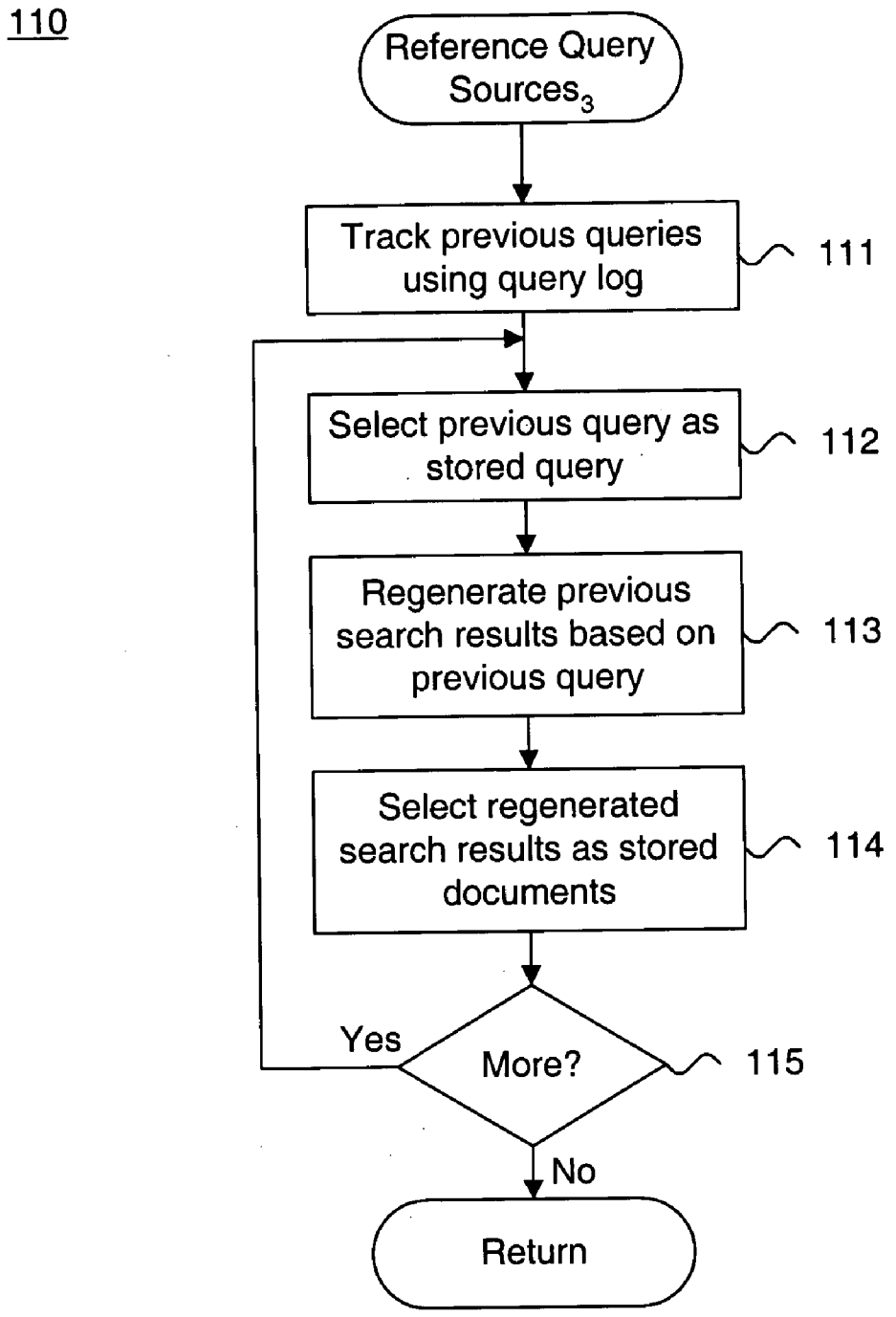


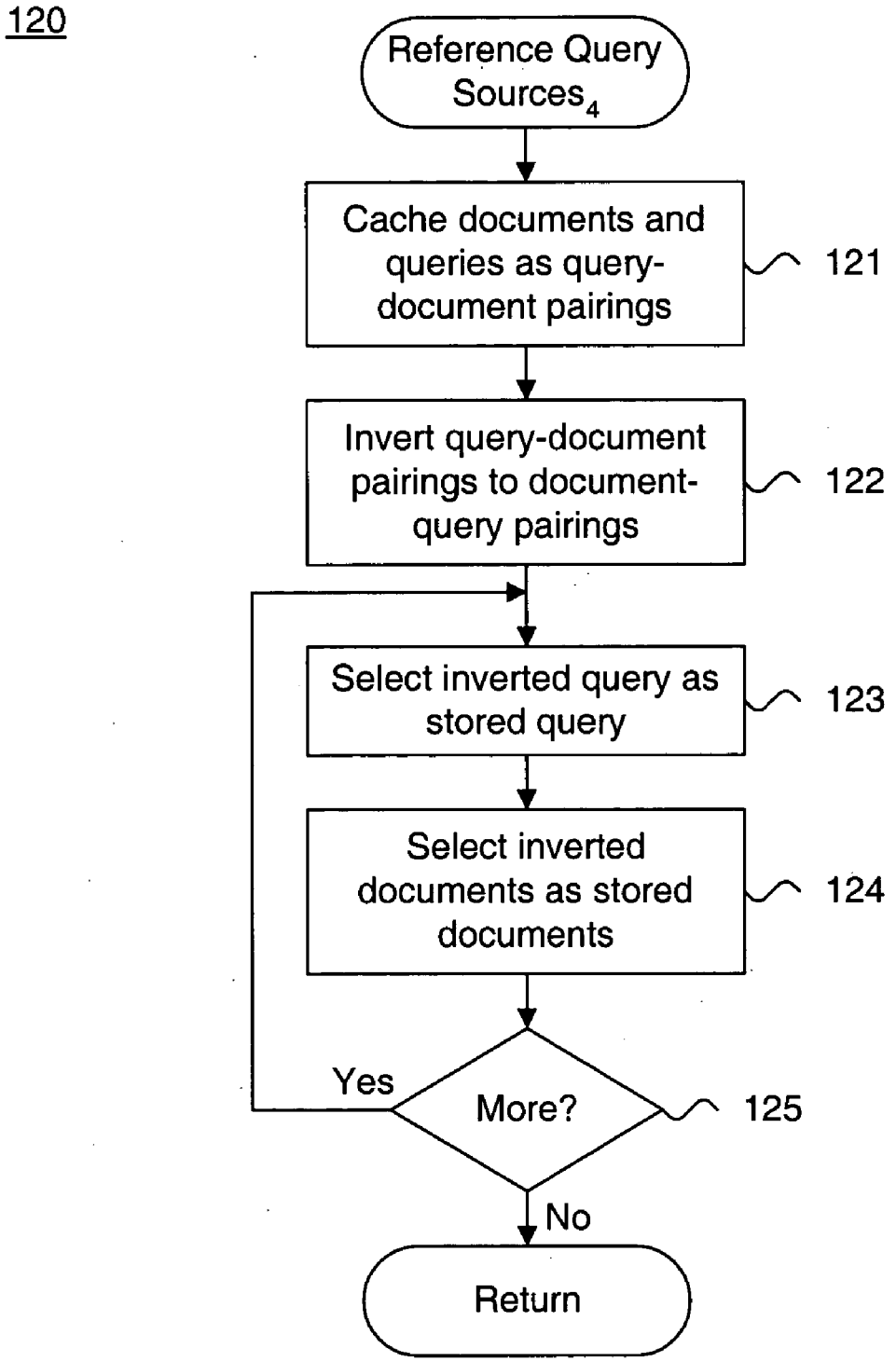
Figure 7.



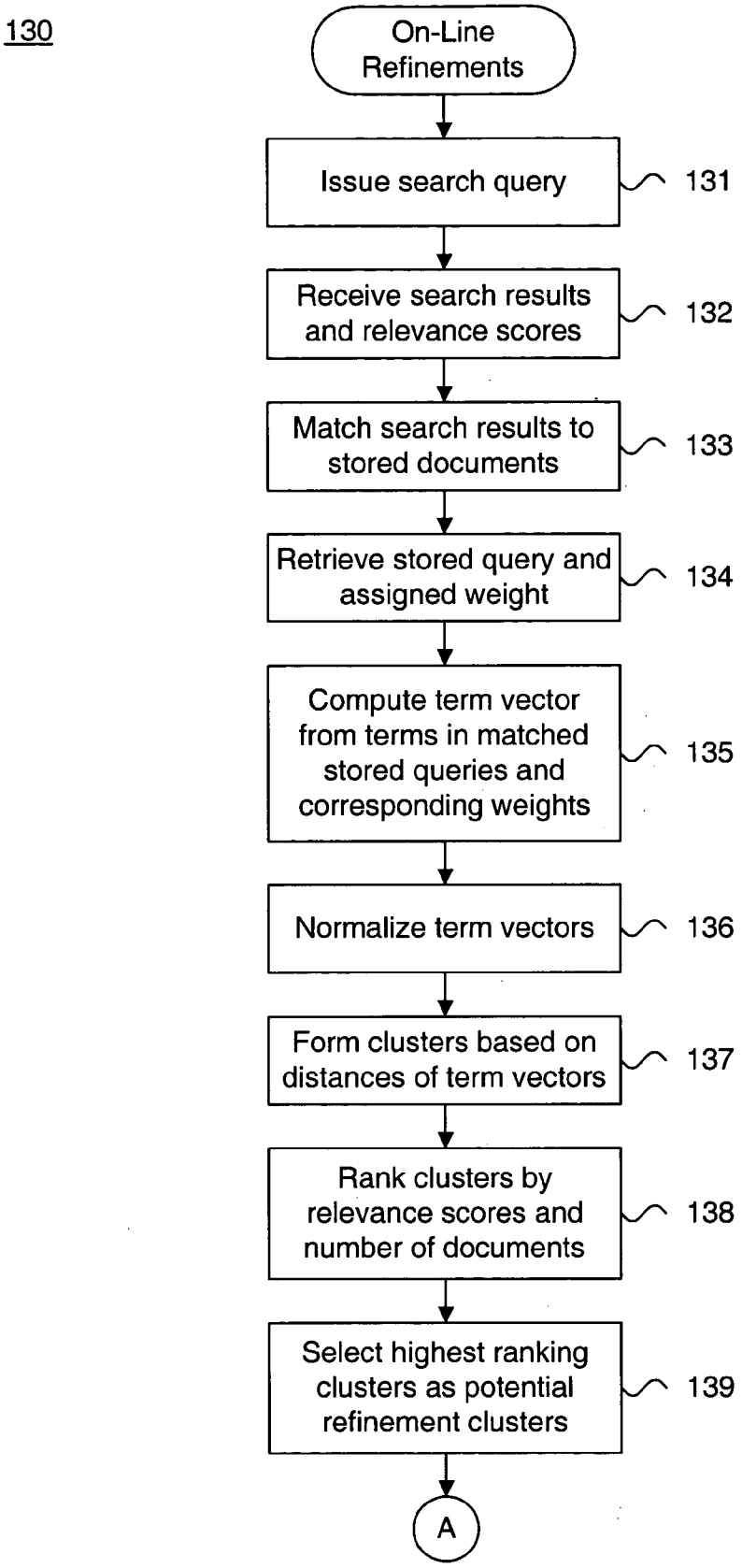
Figur 8.


110
Reference Query Sources3
Track previous queries using query log
111
Select previous query as stored query
112
Regenerate previous search results based on previous query
113
Select regenerated search results as stored documents
114
Yes
More?
115
No
Return

Figur 9.


120
Reference Query Sources4
Cache documents and queries as query-document pairings
121
Invert query-document pairings to document-query pairings
122
Select inverted query as stored query
123
Select inverted documents as stored documents
124
Yes
More?
125
No
Return

Figur 10.


130
On-Line Refinements
Issue search query
131
Receive search results and relevance scores
132
Match search results to stored documents
133
Retrieve stored query and assigned weight
134
Compute term vector from terms in matched stored queries and corresponding weights
135
Normalize term vectors
136
Form clusters based on distances of term vectors
137
Rank clusters by relevance scores and number of documents
138
Select highest ranking clusters as potential refinement clusters
139
A

Figur 10 (Cont).

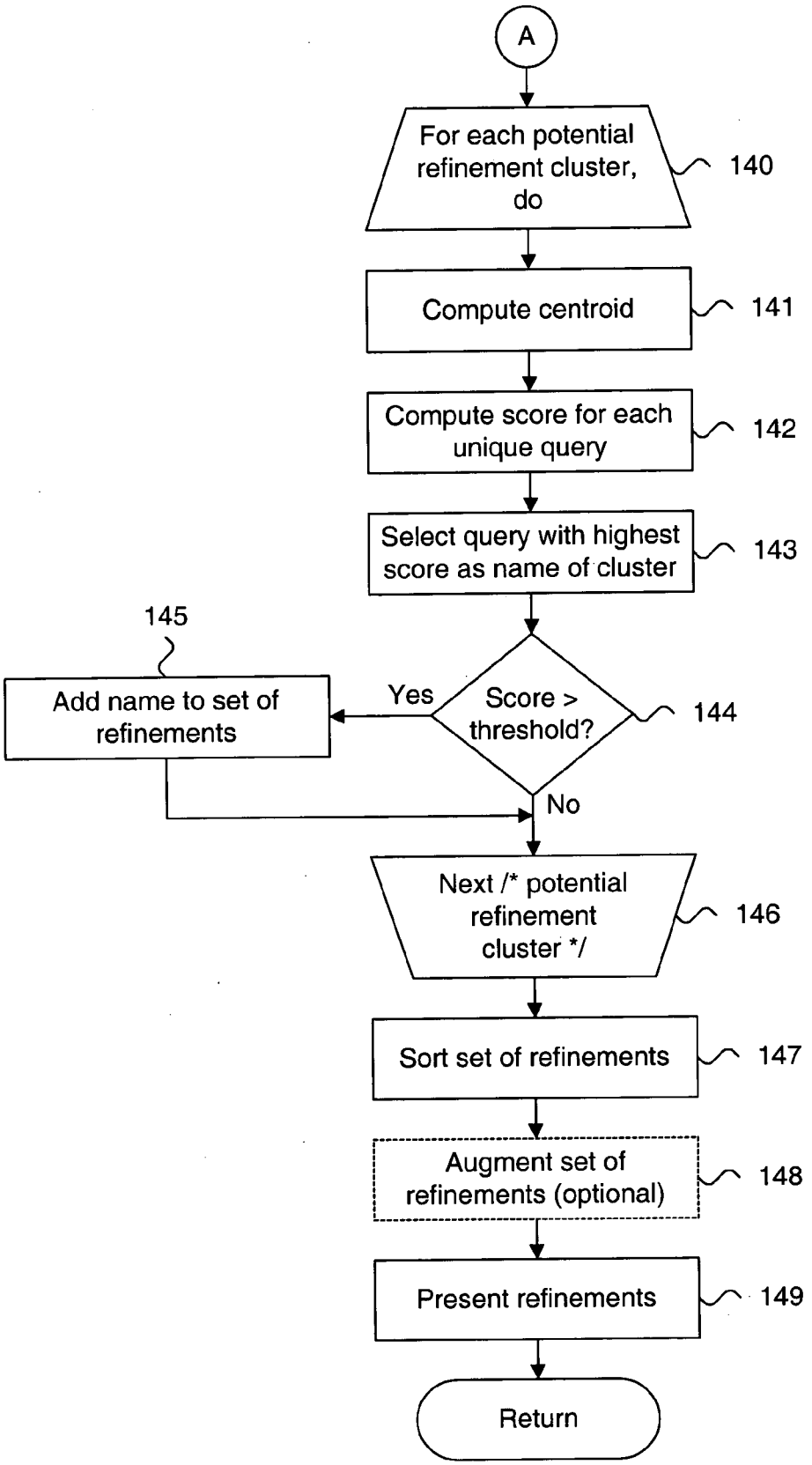


Figure 11.

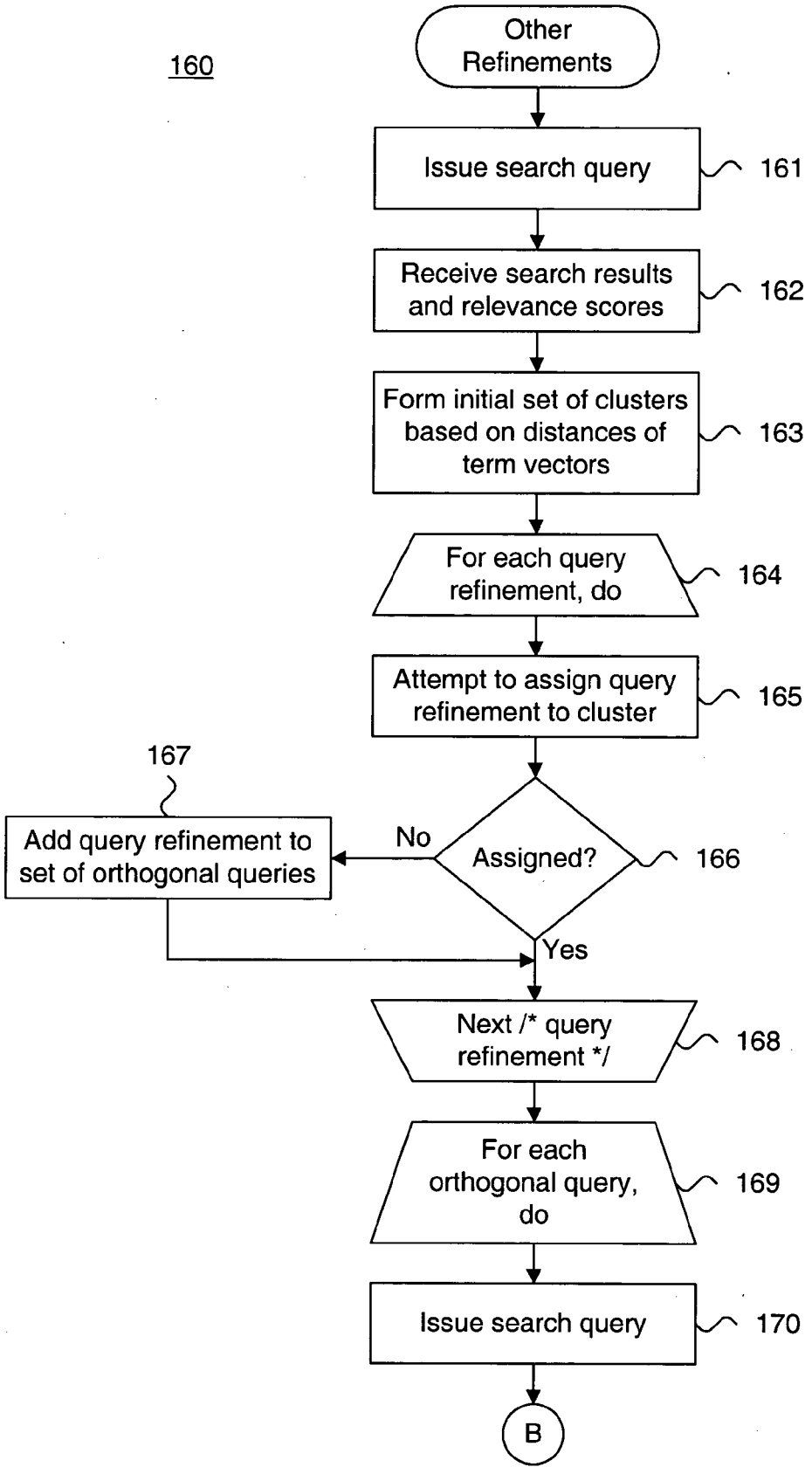
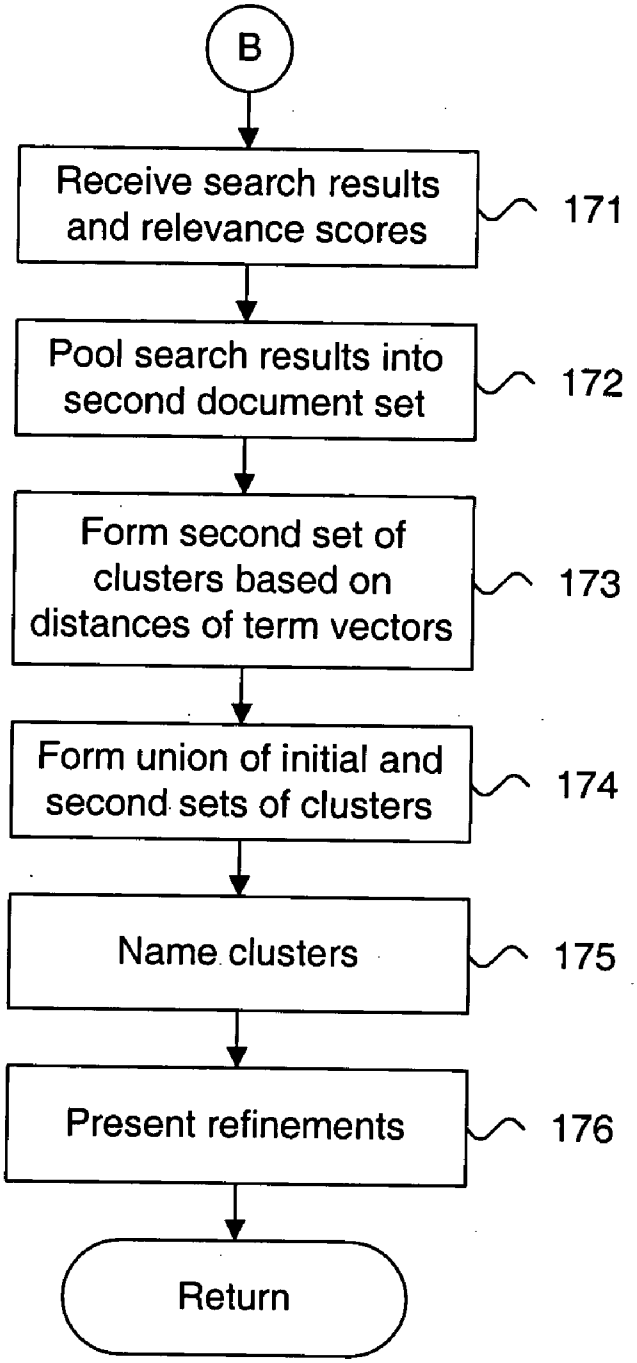

160
Other Refinements
Issue search query
161
Receive search results and relevance scores
162
Form initial set of clusters based on distances of term vectors
163
For each query refinement, do
164
Attempt to assign query refinement to cluster
165
167
Add query refinement to set of orthogonal queries
No
Assigned?
166
Yes
Next /* query refinement */
168
For each orthogonal query, do
169
Issue search query
170
B

Figure 11 (Cont).



SYSTEM AND METHOD FOR PROVIDING SEARCH QUERY REFINEMENTS

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This non-provisional patent application claims priority under 35 USC § 119(e) to U.S. provisional patent application Ser. No. 60/500,539, filed Sep. 5, 2003, the disclosure of which is incorporated by reference.

FIELD OF THE INVENTION

[0002] The present invention relates in general to query processing and, in particular, to a system and method for providing search query refinements.

BACKGROUND OF THE INVENTION

[0003] Although the Internet traces back to the late 1960s, the widespread availability and acceptance of personal computing and internetworking have resulted in the explosive growth and unprecedented advances in information sharing technologies. In particular, the Worldwide Web ("Web") has revolutionized accessibility to untold volumes of information in stored electronic form to a worldwide audience, including written, spoken (audio) and visual (imagery and video) information, both in archived and real-time formats. In short, the Web has provided desktop access to every connected user to a virtually unlimited library of information in almost every language worldwide.

[0004] Search engines have evolved in tempo with the increased usage of the Web to enable users to find and retrieve relevant Web content in an efficient and timely manner. As the amount and types of Web content have increased, the sophistication and accuracy of search engines have likewise improved. Generally, search engines strive to provide the highest quality results in response to a search query. However, determining quality is difficult, as the relevance of retrieved Web content is inherently subjective and dependent upon the interests, knowledge and attitudes of the user.

[0005] Existing methods used by search engines are based on matching search query terms to terms indexed from Web pages. More advanced methods determine the importance of retrieved Web content using, for example, a hyperlink structure-based analysis, such as described in S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Search Engine," (1998) and in U.S. Pat. No. 6,285,999, issued Sep. 4, 2001 to Page, the disclosures of which are incorporated by reference.

[0006] A typical search query scenario begins with either a natural language question or individual terms, often in the form of keywords, being submitted to a search engine. The search engine executes a search against a data repository describing information characteristics of potentially retrievable Web content and identifies the candidate Web pages. Searches can often return thousands or even millions of results, so most search engines typically rank or score only a subset of the most promising results. The top Web pages are then presented to the user, usually in the form of Web content titles, hyperlinks, and other descriptive information, such as snippets of text taken from the Web pages.

[0007] Providing quality search results can be complicated by the literal and implicit scope of the search query itself. A poorly-framed search query could be ambiguous or be too general or specific to yield responsive and high quality search results. For instance, terms within a search query can be ambiguous at a syntactic or semantic level. A syntactic ambiguity can be the result of an inadvertent homonym, which specifies an incorrect word having the same sound and possibly same spelling, but different meaning from the word actually meant. For example, the word "bear" can mean to carry or can refer to an animal or an absence of clothing. A semantic ambiguity can be the result of improper context. For example, the word "jaguar" can refer to an animal, a version of the Macintosh operating system, or a brand of automobile. Similarly, search terms that are too general result in overly broad search results while search terms that are too narrow result in unduly restrictive and non-responsive search results.

[0008] Accordingly, there is a need for an approach to providing suggestions for search query refinements that will resolve ambiguities or over generalities or over specificities occurring in properly framed search queries. Preferably, such an approach would provide refined search queries that, when issued, result in search results closely related to the actual topic underlying the intent of the original search query and provide suggestions that reflect conceptual independence and clear meanings as potential search terms.

SUMMARY OF THE INVENTION

[0009] An embodiment provides a system and method for creating query refinement suggestions. At least one search document retrieved responsive to a query is matched to one or more stored queries. The stored query is scored as a potential query refinement suggestion.

[0010] A further embodiment provides a system and method for providing search query refinements. A stored query and a stored document are associated as a logical pairing. A weight is assigned to the logical pairing. The search query is issued and a set of search documents is produced. At least one search document is matched to at least one stored document. The stored query and the assigned weight associated with the matching at least one stored document are retrieved. At least one cluster is formed based on the stored query and the assigned weight associated with the matching at least one stored document. The stored query associated with the matching at least one stored document are scored for the at least one cluster relative to at least one other cluster. At least one such scored search query is suggested as a set of query refinements.

[0011] A further embodiment provides a system and method for integrating query refinement candidates. At least one search document retrieved responsive to a query is matched to one or more stored documents associated with a stored query and weight. At least one cluster is formed based on the stored query and weight associated with each stored document matched responsive to the query. At least one further search document retrieved responsive to a candidate query is matched to the one or more stored documents. At least one further cluster is formed based on the stored query and weight associated with each stored document matched responsive to the candidate query. The at least one cluster and the at least one further cluster are combined. The stored query for the combined cluster relative to at least one other cluster is scored as a potential query refinement suggestion.

[0012] Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein are described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments and its several details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] **FIG. 1** is a block diagram showing a system for providing search query refinements, in accordance with the present invention.

[0014] **FIG. 2** is a functional block diagram showing a precomputation server, in accordance with the present invention.

[0015] **FIG. 3** is a functional block diagram showing a query refinement server, in accordance with the present invention.

[0016] **FIG. 4** is a flow diagram showing a method for providing search query refinements, in accordance with the present invention.

[0017] **FIG. 5** is a flow diagram showing the routine for performing precomputation for use in the method of **FIG. 4**.

[0018] **FIGS. 6-9** are flow diagrams showing the routines for referencing query sources for use in the method of **FIG. 5**.

[0019] **FIG. 10** is a flow diagram showing the routine for performing on-line query refinements for use in the method of **FIG. 4**.

[0020] **FIG. 11** is a flow diagram showing a routine for integrating candidate query refinements for use in the method of **FIG. 4**, in accordance with a further embodiment.

DETAILED DESCRIPTION

[0021] System Overview

[0022] **FIG. 1** is a block diagram showing a system **10** for providing search query refinements, in accordance with the present invention. A plurality of individual clients **12** are communicatively interfaced to a server **11** via an internetwork **13**, such as the Internet, or other form of communications network, as would be recognized by one skilled in the art. The individual clients **12** are operated by users **19** who transact requests for Web content and other operations through their respective client **12**.

[0023] In general, each client **12** can be any form of computing platform connectable to a network, such as the internetwork **13**, and capable of interacting with application programs. Exemplary examples of individual clients include, without limitation, personal computers, digital assistances, "smart" cellular telephones and pagers, lightweight clients, workstations, "dumb" terminals interfaced to an application server, and various arrangements and configurations thereof, as would be recognized by one skilled in the art. The internetwork **13** includes various topologies, configurations, and arrangements of network interconnectivity components arranged to interoperatively couple with enterprise, wide area and local area networks and include, without limitation, conventionally wired, wireless, satellite, optical, and equivalent network technologies, as would be recognized by one skilled in the art.

[0024] For Web content exchange and, in particular, to transact searches, each client **12** executes a Web browser **18** ("Web browser"), which preferably implements a graphical user interface and through which search queries are sent to a Web server **20** executing on the server **11**. Each search query describes or identifies information, generally in the form of Web content, which is potentially retrievable via the Web server **20**. The search query provides search characteristics, typically expressed as individual terms, such as keywords and the like, and attributes, such as language, character encoding and so forth, which enables a search engine **21**, also executing on the server **11**, to identify and send back search result documents, generally in the form of Web pages. Other styles, forms or definitions of search queries and characteristics are feasible, as would be recognized by one skilled in the art.

[0025] The Web pages are sent back to the Web browser **18** for presentation, usually in the form of Web content titles, hyperlinks, and other descriptive information, such as snippets of text taken from the Web pages. The user can view or access the Web pages on the graphical user interface and can input selections and responses in the form of typed text, clicks, or both. The server **11** maintains a search database **15** in which Web content **22** is maintained. The Web content **22** could also be maintained remotely on other Web servers (not shown) interconnected either directly or indirectly via the internetwork **13** and which are preferably accessible by each client **12**. In a further embodiment, the server **11** maintains a cache **23** in which cached documents **24** and cached queries **25** are maintained. The cache **23** associates each cached document **24** with one or more cached queries **25** to improve searching performance, as is known in the art. Finally, in a still further embodiment, the search engine **21** maintains a query log **26** in which records of previous search queries **27** are tracked.

[0026] The search engine **21** preferably identifies the Web content **22** best matching the search characteristics to provide high quality Web pages, such as described in S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Search Engine" (1998) and in U.S. Pat. No. 6,285,999, issued Sep. 4, 2001 to Page, the disclosures of which are incorporated by reference. In identifying matching Web content **22**, the search engine **21** operates on information characteristics describing potentially retrievable Web content. Note the functionality provided by the server **20**, including the Web server **20** and search engine **21**, could be provided by a loosely- or tightly-coupled distributed or parallelized computing configuration, in addition to a uniprocessing environment.

[0027] Search queries can potentially be ambiguous or lack generality or specificity. Such poorly-framed search queries can be remedied through search query refinements, which can be provided in response to search query issuances. Search query refinements are generated and suggested as a two-part operation. First, a search query is associated with a search result document in a one-to-one association and a weight is assigned to each search query-

search result document association, as further described below with reference to **FIG. 2**. Second, the search query-search result document associations and assigned weights are matched to at least one new search query to form scored clusters, as further described below with reference to **FIG. 3**. The clusters are projected from multi-dimensional space based on constituent query terms and unique search queries associated with any document in each cluster are scored. The clusters are named based on the scoring, preferably by highest scores. The named clusters are sorted and provided as suggested refinements to the original search query.

[**0028**] The individual computer systems, including server **11** and clients **12**, include general purpose, programmed digital computing devices consisting of a central processing unit (processors **13** and **16**, respectively), random access memory (memories **14** and **17**, respectively), non-volatile secondary storage, such as a hard drive or CD ROM drive, network or wireless interfaces, and peripheral devices, including user interfacing means, such as a keyboard and display. Program code, including software programs, and data is loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage. The Web browser **18** is an HTTP-compatible Web browser, such as the Internet Explorer, licensed by Microsoft Corporation, Redmond, Wash.; Navigator, licensed by Netscape Corporation, Mountain View, Calif.; or other forms of Web browsers, as are known in the art.

[**0029**] Precomputation Server

[**0030**] **FIG. 2** is a functional block diagram **30** showing a precomputation server **34**, in accordance with the present invention. The precomputation system **31** builds a set of associated queries and documents, preferably as an off-line operation. The precomputation system **31** includes a precomputation engine **34**, which associates a stored query **40** with a stored document **41** and assigns a weight **43** to each association **42**, as further described below with reference to **FIG. 5**.

[**0031**] The precomputation system **31** builds and maintains the association database **39**. The association database **39** stores the stored queries **40**, stored documents **41**, associations **42**, and weights **43**, which are used by a query refinement server to formulate and suggest query refinements, as further described below with reference to **FIG. 3**. In further embodiments, the precomputation system **31** also references the query log **26**, which is stored in the search database **15**, and the cached documents **24** and cached queries **25**, which are stored in the cache (both shown in **FIG. 1**).

[**0032**] The precomputation engine **34** logically includes four modules. Other logical arrangements and definitions of functional modules are possible, as would be recognized by one skilled in the art. First, an associator **35** associates a stored query **40** with a stored document **41** and the stored query-stored document pairing is maintained in the association database **39** as a set of associations **42**. Each stored query **40** is associated with only one stored document **41**, although any given stored query **40** can be paired with one or more stored documents **41** in a one-to-many relationship. Each individual pairing is maintained as a separate association **42** in the association database **39**. However, the individual pairings need not be explicitly stored in the form of associations **42** and can instead be logically recorded or tracked, such as by using a mapping, table or other means for matching stored queries **40** with stored documents **41**, as would be recognized by one skilled in the art.

[**0033**] The associator **35** also assigns a weight **43** to each association **42**. Each weight **43** estimates the relevance of the stored document **41** to the associated stored query **40**. When multiple associations **42** for the same stored query-stored document pairing occurs, the weights **43** for each of the multiple associations **42** are summed. If query frequency data is available, such as, for example, when a search query **59** is cached, the weight **43** is multiplied by the search query frequency, that is, the number of times that the search query **59** has been issued.

[**0034**] Second, a selector **36** selects one or more of the stored documents **41** for association with a stored query **40** based on an issued search. In the described embodiment, the selector **36** selects the stored documents **41**, which are each separately associated with a stored query **40** based on either a search document chosen following an issued search or from a set of search results received for an issued search, as further described below respectively with reference to **FIGS. 6 and 7**.

[**0035**] Third, the regenerator **37** selects one or more of the stored documents **41**, which are each separately associated with a stored query **40** based on the query log **26**. In a further embodiment, the regenerator **37** regenerates the search results from previously tracked queries **27**, as reflected in the query log **26**. The regenerator **37** selects the regenerated search results as stored documents **41**, which are each separately associated with a previously tracked search query **27**, as further described below with reference to **FIG. 8**.

[**0036**] Fourth, the inverter **38** selects one or more of the stored documents, which are each separately associated with a stored query **40** based on cached data. In a still further embodiment, the inverter **38** evaluates the cached documents **24** and cached queries **25** and inverts the cached document-cached queries pairings into cached query-cached documents pairings. The inverter **38** selects the inverted cached documents **24** as stored documents **41**, which are each separately associated with a cached query **25**, as further described below with reference to **FIG. 9**.

[**0037**] The individual computer system, including the precomputation system **31**, include general purpose, programmed digital computing devices consisting of a central processing unit (processor **33**), random access memory (memory **32**), non-volatile secondary storage, such as a hard drive or CD ROM drive, network or wireless interfaces, and peripheral devices, including user interfacing means, such as a keyboard and display. Program code, including software programs, and data is loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage. Note the functionality provided by the precomputation system **31** could be provided by a loosely- or tightly-coupled distributed or parallelized computing configuration, in addition to a uniprocessing environment.

[**0038**] Query Refinement Server

[**0039**] **FIG. 3** is a functional block diagram **50** showing a query refinement system **51**, in accordance with the present invention. The query refinement system **51** formulates and

suggests one or more query refinements **67**. The query refinements can be formulated either as an on-line operation following a search query issuance or based on precomputations for a given set of search queries. The query refinement system **51** includes a query refinement engine **54**, which formulates the query refinements **67** as suggestions in response to an actual search query **59**, as further described below with reference to **FIG. 10**, and which integrates candidate query refinements **68**, as further described below with reference to **FIG. 11**.

[**0040**] The query refinement engine **54** logically includes four modules. Other logical arrangements and definitions of functional modules are possible, as would be recognized by one skilled in the art. First, a matcher **55** matches one or more of the stored documents **41** to the actual search documents **60**, generated by the search engine **21** (shown in **FIG. 1**) in response to the issuance of a search query **59**. The search engine **21** also generates relevance scores **60** as part of the search query issuance. The matcher **55** identifies the stored queries **40** and assigned weights **43** using the associations **43** corresponding to the matched stored documents **41**.

[**0041**] Second, a clusterer **56** forms one or more clusters **62** based on term vectors **62** formed from the terms occurring in the matched stored queries **40** and corresponding weights **43**. The term vectors **62** are normalized vectors projected into multi-dimensional space, with each dimension corresponding to a term, which can be an individual word or word combination. The clusters **62** are ranked based on the relevance scores **61** assigned to the search documents **60** corresponding to the matched stored documents **41** and the number of stored documents **41** occurring in each cluster **62**. The highest ranking clusters **62** are selected as potential refinement clusters **64**. In the described embodiment, the clusters **62** are formed using a hierarchical agglomerative clustering algorithm, such as described in E. Rasmussen, "Clustering Algorithms," in "Information Retrieval," (W. Frakes & R. Baeza-Yates eds. 1992), the disclosure of which is incorporated by reference, although other types of clustering algorithms could be used, as would be recognized by one skilled in the art.

[**0042**] Third, a scorer **57** computes center-weighted term vectors, referred to as centroids **65**, which each represent the weighted center of the term vector **62** of each cluster **63**. The centroids **65** are computed from each of the potential refinement clusters **64**. The scorer **57** assigns scores **66** to each unique search query in each of the potential refinement clusters **64** based on the number of stored documents **41** with which the search query is associated and the distance from the centroid **65**. Other approaches to computing centroids **65** could also be used, including using unweighted values and by varying the forms of weighting and averaging, as would be recognized by one skilled in the art.

[**0043**] Finally, a presenter **58** identifies the substantially highest scoring search queries as one or more query refinements **67** to the user **19**. Importantly, the details of the formulation of the search query refinements, including the term vectors **62**, clusters **63**, potential refinement clusters **64**, centroids **65**, and scores **66**, are encapsulated by the query refinement engine **54**. Thus, a user **19** will only be aware of the actual suggested query refinements **67**.

[**0044**] The individual computer system, including the query refinement system **51**, include general purpose, programmed digital computing devices consisting of a central processing unit (processor **53**), random access memory (memory **52**), non-volatile secondary storage, such as a hard drive or CD ROM drive, network or wireless interfaces, and peripheral devices, including user interfacing means, such as a keyboard and display. Program code, including software programs, and data is loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage. Note the functionality provided by the query refinement system **51** could be provided by a loosely- or tightly-coupled distributed or parallelized computing configuration, in addition to a uniprocessing environment.

[**0045**] Method Overview

[**0046**] **FIG. 4** is a flow diagram showing a method **70** for providing search query refinements, in accordance with the present invention. The method **70** is described as a sequence of process operations or steps, which can be executed, for instance, by the precomputation engine **34** of **FIG. 2** and the query refinement engine **54** of **FIG. 3**, or equivalent components.

[**0047**] The method **70** performs precomputation (Block **71**) and query refinement (Block **72**). Precomputation (Block **71**) builds the association database **39** (shown in **FIG. 2**) by associating and storing representative stored queries **40** with stored documents **41**, along with associations **42** and weights **43**, as further described below with reference to **FIG. 5**. In the described embodiment, precomputation is performed as an off-line operation independent of any specific search query.

[**0048**] Query refinement (Block **72**) formulates query refinement suggestions structured to better frame search queries posted by users. In one embodiment, the query refinement suggestions are performed as an on-line operation based on search query results observed for issued search queries, which can be matched and applied to the data maintained in the association database **39**, as further described below with reference to **FIG. 10**. In a further embodiment, candidate query refinements **68** can be integrated into query refinement suggestions, which can also be matched and applied to the data maintained in the association database **39**, as further described below with reference to **FIG. 11**.

[**0049**] The method terminates upon the completion of the last query refinement operation.

[**0050**] Precomputation Routine

[**0051**] **FIG. 5** is a flow diagram showing the routine **80** for performing precomputation for use in the method **70** of **FIG. 4**. The purpose of this routine is to build the association database **39** (shown in **FIG. 3**) and associate stored queries **40** with stored documents **41** to form associations **42** and assign weights **43**.

[**0052**] Initially, query sources are referenced to build the stored queries **40** and stored documents **41** maintained in the association database **39**. Briefly, query source referencing refers to identifying appropriate pairings of search queries and search result documents based on actual search query issuances, including current, stored and cached search query issuances. Thus, the stored queries **40** can originate from actual search queries, as further described below with ref-

erence to **FIGS. 6 and 7**, previous search queries **27** tracked in the query log **26**, as further described below with reference to **FIG. 8**, and cached queries **25** maintained in the cache **13**, as further described below with reference to **FIG. 9**. In addition, the stored documents **41** can be search documents **60** retrieved in response to actual search queries **59**, or can be regenerated search results or cached documents **24** maintained in the cache **13**. Other sources of search queries and documents for association as stored queries **40** and stored documents **41** are possible, as would be recognized by one skilled in the art.

[**0053**] Once the stored queries **40** for the association database **39** have been identified and stored, each stored query **40** is iteratively processed (Block **82**), as follows. A stored query **40** is separately associated with each stored document **41** (Block **83**). A one-to-one association **42** is formed, provided, however, that each associated stored document **41** is based on the issuance of the corresponding associated stored query **40**. Each stored query **40** is separately associated with only one stored document **41**, although any given stored query **40** can be paired with one or more stored documents **41** in a one-to-many relationship. A weight **43** is assigned to the association **42** (Block **84**), reflecting the relevance of the stored document **41** to the stored query **40**. Processing continues with each remaining stored query **40** (Block **85**), after which the routine returns.

[**0054**] Query Source Referencing Routines

[**0055**] **FIGS. 6-9** are flow diagrams showing the routines for referencing query sources for use in the method **80** of **FIG. 5**. The purposes of these routines are to populate the stored queries **40** and stored documents **41** maintained in the association database **39**. Each of the routines identifies search queries **59** and related search documents **60**, respectively for use as stored queries **40** and stored documents **41**.

[**0056**] Each stored query **40** in an association **42** is associated with a stored document **41**, although any given stored query **40** can be paired with one or more stored documents **41** in a one-to-many relationship. Each individual pairing is maintained as a separate association **42**. However, the individual pairings need not be explicitly stored in the form of associations **42** and can be logically recorded or tracked, such as by using a mapping, table or other means for matching stored queries **40** with stored documents **41**, as would be recognized by one skilled in the art.

[**0057**] In the association database, each of the stored queries **40** take the form of a search query **59** expressed as, for instance, keywords or terms. Terms include individual words or combinations of words. Each of the stored documents **41** are preferably stored as references and not as actual content. Generally, each stored document **41** can be in the form of, by way of example, a uniform resource locator (URL), hyperlink, anchor, or document excerpt.

[**0058**] Query Source Referencing Using a Chosen Search Document

[**0059**] **FIG. 6** is a flow diagram showing a routine **90** for performing precomputation based on a chosen search document. The purpose of this routine is to identify a one-to-one pairing relationship between a search query **59** and a selected search document **60**. The one-to-one relationship can occur, for instance, when a user **19** selects a particular search document **60** from among a set of search results following the issuance of a search query **59**.

[**0060**] First, a search query **59** is issued (Block **91**) and search results, consisting of search documents **60** (shown in **FIG. 3**), are received (Block **92**). The actual search query **59** is selected as the stored query **40** (Block **93**). A search document **59** is selected as a stored document **41**, when the search document **59** is chosen by the user **19** performing the search from among the search results (Block **94**). The routine then returns.

[**0061**] Query Source Referencing Using Search Results

[**0062**] **FIG. 7** is a flow diagram showing a routine **100** for performing precomputation based on a search results. The purpose of this routine is to identify a one-to-many pairing relationship between a search query **59** and a set of search results **60**. The one-to-many relationship occurs when a set of search results are identified following the issuance of a search query **59**.

[**0063**] The search is query issued (Block **101**) and search results, consisting of search documents **60** (shown in **FIG. 3**), are received (Block **102**). The actual search query **59** is selected as the stored query **40** (Block **103**). The set of search documents **59** are selected as stored documents **41** (Block **104**). Note that each search query is separately associated with only one search document and multiple search query-search document associations can be formed for any given search query. The routine then returns.

[**0064**] Query Source Referencing Using Tracked Queries

[**0065**] **FIG. 8** is a flow diagram showing a routine **110** for performing precomputation based on previously tracked queries. The purpose of this routine is to reference the query log **26** (shown in **FIG. 1**) for previous search queries **27** and to regenerate search results based on those previously tracked queries **27**.

[**0066**] Initially, previous search queries **27** are tracked using the query log **26** (Block **111**). Each previous search query **27** is selected as a stored query **40** (Block **112**) and search results based on the previous search query are regenerated (Block **113**). The regenerated search results are selected as stored documents **41** (Block **114**). Note that each previous search query is separately associated with only one regenerated search result document and multiple previous search query-regenerated search result document associations can be formed for any given previous search query. If further previous search queries **27** remain in the query log **26** (Block **115**), processing continues with the next previous search query **27** (Block **112**). Otherwise, the routine returns.

[**0067**] Query Source Referencing Using Cached Data

[**0068**] **FIG. 9** is a flow diagram showing a routine **120** for performing precomputation based on cached documents and queries. The purpose of this routine is to invert pairings of cached document-cached queries maintained in the cache **23** for use as stored queries **40** and stored documents **41**.

[**0069**] Cached documents **24** and cached queries **25** are maintained in the cache **23** (shown in **FIG. 1**). The cached documents **24** and cached queries **25** are organized in the cache **23** as cached query-cached document pairings. However, the associations **42** and weights **43** (shown in **FIG. 3**) are based on stored document-stored query pairings. Thus,

the cached query-cached document pairings implicit in the organization of the cache **23** are inverted to form cached document-cached query pairings (Block **122**). An inverted search query is selected as a stored query **40** (Block **123**) and the corresponding inverted documents are selected as stored documents **41** (Block **124**). Note that each inverted search query is separately associated with only one inverted document and multiple inverted search query-inverted document associations can be formed for any given inverted search query. If further cached query-cached documents pairings remain (Block **125**), processing continues with the selection of the next inverted pairing (Block **123**). Otherwise, if no further cached document-cached queries pairings remain (Block **125**), the routine returns.

[**0070**] On-Line Query Refinements Routine

[**0071**] **FIG. 10** is a flow diagram showing the routine **130** for performing on-line query refinements for use in the method **80** of **FIG. 4**. The purpose of this routine is to formulate one or more search query refinements **67** preferably on-line, which can be suggested following an actual search query **59** issuance.

[**0072**] Initially, a search query **59** is issued (Block **131**) and search results, in the form of search documents **60**, and relevance scores **61** are received (Block **132**). If possible, the stored documents **41** are matched to the search results (Block **133**). Ideally, at least one of the search results will match a stored document **41**. However, as the association database **39** is preferably built as an off-line operation, the set of stored documents **41** may not fully match every possible search results. Accordingly, those search results, which do not have a matching stored document **41**, are skipped.

[**0073**] Next, for each matched search result, the association **42** corresponding to the matched stored document **41** is determined and is used to retrieve the associated stored queries **40** and weights **43** (Block **134**). A term vector **62** is then computed from the terms occurring in the matched stored queries **40** and corresponding weights **43** (Block **135**). Each term vector **62** is a vector in multi-dimensional space, where each dimension corresponds to a distinct term and each term represents an individual word or word combination. The length of a term vector **62** in each dimension equals the sum of the weights of the corresponding term in the set of associated queries. Those term vector elements corresponding to the terms from the original search query **59** are multiplied by a constant factor to downwardly weight the terms to enforce independence from the original search query **59**. The term vectors **62** are normalized (Block **136**). In the described embodiment, the term vectors **62** are length normalized to a length of one, although other normalizations are possible, as would be recognized by one skilled in the art.

[**0074**] Clusters **63** are then formed based on the distances of the term vectors **62** from a common origin (Block **137**). In the described embodiment, the clusters **62** are formed using a hierarchical agglomerative clustering algorithm, such as described in E. Rasmussen, described supra., the disclosure of which is incorporated by reference, although other forms of clustering could also be applied, as would be recognized by one skilled in the art.

[**0075**] The resulting clusters **63** are ranked using the relevance scores **61** assigned to the search documents **60** corresponding to the matched stored documents **41** and the number of stored documents **41** occurring in each cluster **63** (Block **138**). The highest ranking clusters are selected as the potential refinement clusters **64** (Block **139**). In the described embodiment, the potential refinement clusters **63** are selected based on a predefined threshold value, although other cluster selection criteria are possible, as would be recognized by one skilled in the art.

[**0076**] For each potential refinement cluster **64** (Block **140**), a centroid **65** is computed (Block **141**). Each centroid **65** represents the weighted center of the term vector **62** for each cluster **63**, as a normalized sum of the product of the term vector **62** for each stored query **40** and the relevance score **61** assigned to the search documents **60** corresponding to the matched stored documents **41** of the original search query **59**. Other approaches to computing centroids **65** could also be used, including using unweighted values and by varying the forms of weighting and averaging, as would be recognized by one skilled in the art.

[**0077**] A score **66** is then computed for each unique search query **59** occurring in the potential refinement cluster **64** (Block **143**). Each score **66** is computed as the product of the frequency of the stored query **40** for the cluster **63** times the length of the distance vector measured from the term vector **62** of the stored query **40** to the centroid **65** of the cluster **63**. Other forms of scoring, ordering and ranking are possible, as would be recognized by one skilled in the art. The stored query **40** with the highest score **66** is selected as the name of the cluster **63** (Block **143**). Alternatively, other cluster naming selection criteria using highest, averaged, lowest, or other forms of scoring, ordering and ranking are possible, as would be recognized by one skilled in the art. If the score **66** for the unique stored query **40** exceeds a predefined threshold (Block **144**), the name is added to the set of query refinements **67** (Block **145**). Processing continues with each remaining potential refinement cluster **64** (Block **146**).

[**0078**] Finally, the set of refinements **67** are sorted into rankings (Block **147**) as a function of the relevance scores **61** assigned to the search documents **60** corresponding to the matched stored documents **41** appearing in each cluster **63** plus the size of the cluster **63** in number of stored documents **41**. As an optional step, the set of refinements **67** are augmented with supplemental queries (Block **148**). In the described embodiment, each supplemental query consists of the terms originally appearing in the search query **59** and negated forms of all terms appearing in the set of refinements **67**, but not appearing in the original search query. Other forms of augmenting the set of refinements **67** are feasible, as would be recognized by one skilled in the art. The set of refinements **67** are presented (Block **149**) based on the rankings and, optionally, cluster scores. The routine then returns.

[**0079**] Integrating Candidate Query Refinements Routine

[**0080**] **FIG. 11** is a flow diagram showing a routine **160** for integrating candidate query refinements for use in the method **80** of **FIG. 4**, in accordance with a further embodiment. The purpose of this routine is to formulate one or more search query refinements **67**, which can be suggested by integrating candidate query refinements **68**. The candidate query refinements **68** can originate from any external source, as would be recognized by one skilled in the art, including the user **19** as user-specified query refinements, third parties,

and other approaches to attempting to remedy poorly-framed search queries. For brevity, those detailed operations previously presented above with reference to **FIG. 10** are described in summary fashion where applicable.

[**0081**] By way of example, other approaches attempt to remedy poorly-framed search queries by suggesting refinements, which can be considered and selected by a user as an adjunct to or in lieu of the original search query. One approach suggests previous, recurring search queries, which contain the terms used in the original search query, along with other terms. However, the suggested queries can overlap in meaning with the original search query and word co-occurrences and frequencies poorly divide the search space into conceptually independent partitions. Another related approach tracks search query refinements entered by other users and suggests frequently-used refinements to modify the original search query. However, reliance on only frequently occurring refinements can also poorly divide the search space into conceptually independent partitions. Yet another related approach clusters documents received in response to an original search query based on the terms occurring in each document. The terms that characterize each of the clusters are used as cluster names and suggested as refinements. However, the resulting refinements often consist of terms not generally occurring in a user-specified search query and, as a result, can be difficult to understand and can perform poorly when used as a search query.

[**0082**] Initially, as before, a search query **59** is issued (Block **161**) and search results, in the form of search documents **60**, and relevance scores **61** are received (Block **162**). Following search results matching and term vector computation and normalization, an initial set of clusters **63** is formed based on the distances of the term vectors **62** (Block **163**). For each candidate query refinement **68** (Block **164**), an attempt is made to assign the candidate query refinement **68** to one of the clusters **63** (Block **165**). If the candidate query refinement **68** is not assigned to a cluster **63** (Block **166**), the candidate query refinement **68** is added to a set of orthogonal queries (Block **167**). Processing continues with each remaining candidate query refinement **68** (Block **168**).

[**0083**] Next, for each candidate query refinement **68** in the set of orthogonal queries (Block **169**), a search query **59** is issued (Block **170**) and search results, in the form of search documents **60**, and relevance scores **61** are again received (Block **171**). The search results are pooled into a second set of documents (Block **172**). Following search results matching and term vector computation and normalization, a second set of clusters **63** is formed based on the distances of the term vectors **62** (Block **173**). A union is formed of the initial and second sets of clusters **63** (Block **174**). Following cluster ranking, potential refinement cluster **64** selection, centroid **65** computation, and unique query scoring, the clusters **63** in the union are named for the unique queries with the highest scores (Block **175**). Finally, the set of refinements **67** are presented (Block **149**) based on the rankings and, optionally, cluster scores. The routine then returns.

[**0084**] Query Refinement Example

[**0085**] By way of example, a user **19** might submit a search query **59**, which includes the individual word, "jaguar." Upon issuance, a set of search documents **60** are received and the top **100** documents are chosen for clustering. Note a set of 100 documents is used merely for the purposes of illustration and any other limit would be equally suitable, as would be recognized by one skilled in the art. The search documents **60** might naturally identify several relevant semantic groupings, including documents about automobiles manufactured by Jaguar Corporation, including hyperlink references to the official Jaguar Corporation Websites in the United States and United Kingdom and a Jaguar brand automobile owners association. The semantic groupings might also include documents about the Macintosh operating system version code-named jaguar, documents about jaguar animals, as well as documents about a number of other miscellaneous topics that may not be groupable into cohesive document clusters **63**.

[**0086**] During the clustering phase, term vectors **62** are computed for each of the top 100 documents ranked by relevance score. As above, a set of 100 documents is used merely for the purposes of illustration and any other limit would be equally suitable, as would be recognized by one skilled in the art. Each selected search document **60** is matched to a stored document **41** in the association database **39** and the corresponding stored queries **40** are determined by looking up the associations **42** for each matched stored document **41**. The term vectors **62** are formed by flattening the constituent terms for each corresponding stored query **40** into a simple vector space. Clusters **63** are generated from the term vectors **62**, which typically extracts the relevant semantic groupings, such as those groupings described above.

[**0087**] A cluster centroid **65** is calculated for each cluster **63**. All search queries **59** associated with a search document **60** in the cluster **63** are scored according to the distance from the cluster centroid **65** and the percent of stored documents **41** occurring in the cluster **63** with which each stored document **41** is associated. For instance, assume that a cluster **63** is calculated for the example "jaguar" search query **59** for the semantic grouping containing documents about Jaguar brand automobiles. In the corresponding cluster centroid **65**, the dominant terms include words, such as "jaguar,""automobile,""auto,""car,""USA,""UK," and so forth. The best matching query name suitable as a suggested query refinement **67** would be "jaguar car," which has good coverage over the entire cluster **63** and also contains the two terms having a highest weight in the cluster centroid **65**.

[**0088**] Similarly, further assume that a cluster **63** is calculated for the semantic grouping containing documents about the Macintosh code-named jaguar operating system. In the corresponding cluster centroid **65**, the dominant terms include words, such as "jaguar,""X,""Mac,""OS," and so forth. The best matching query name suitable as a suggested query refinement **67**, assuming case insensitivity, would be "mac os x jaguar," which contains all of the top search query terms and appears in many of the documents in the cluster **63**. Other generated clusters **63** and query names suitable as suggested query refinement **67** include "jaguar racing" for documents about Jaguar automobile racing clubs and "jaguar cat" for documents about the jaguar animal.

[**0089**] Finally, the refinements **67** are sorted as a function of the relevance scores **61** assigned to the search documents **60** corresponding to the matched stored documents **41** appearing in each cluster **63** plus the size of the cluster **63** in number of stored documents **41**. A cluster **63** will be

ranked higher than another cluster **63** if the cluster **63** is either larger or has stored documents **41** having higher relevance scores **61**. In the example, the final ranking of the refinements **67** includes "jaguar car,""mac os x jaguar, ""jaguar racing," and "jaguar cat." The rankings and, optionally, cluster scores are used for presentation purposes.

[**0090**] In a further embodiment, the refinements **67** would include negated forms of all terms appearing in the set of refinements **67**, but not appearing in the original search query. Thus, the alternative refinements **67** include "jaguar -car -mac-os-x -racing -cat." In a still further embodiment, a predetermined set of search queries **59** selected from past user queries could be used to precompute possible sets of refinements **67** for the predetermined queries. The predetermined queries would be issued and the search results would be maintained in a database for look up in response to user search requests based on the predetermined queries.

[**0091**] While the invention has been particularly shown and described as referenced to the embodiments thereof, those skilled in the art will understand that the foregoing and other changes in form and detail may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A system for creating query refinement suggestions, comprising:

a matcher matching at least one search document retrieved responsive to a query to one or more stored queries; and

a scorer scoring the stored query as a potential query refinement suggestion.

2. A system according to claim 1, further comprising:

a document matcher matching the at least one search document to one or more stored documents associated with the one or more stored queries.

3. A system according to claim 1, further comprising:

a weight associated with at least one such stored query; and

a clusterer forming at least one cluster based on the stored query and weight.

4. A system according to claim 3, further comprising:

a term vector used in cluster formation computed from terms extracted from the stored query and based on the weight for the stored query.

5. A system according to claim 4, further comprising:

a distance for the term vector; and

the cluster forming the at least one cluster relative to the distance.

6. A system according to claim 3, further comprising:

a ranker ranking the at least one cluster relative to the at least one other cluster by evaluating a relevance score associated with each search document corresponding to each matched stored document.

7. A system according to claim 6, further comprising:

a selector selecting at least one ranked cluster as a potential refinement cluster.

8. A system according to claim 3, further comprising:

a centroid computed as a weighted center of the at least one cluster; and

a score computed for each stored query relative to the centroid.

9. A system according to claim 8, further comprising:

a selector naming the at least one cluster for at least one scored stored query.

10. A system according to claim 1, further comprising:

a threshold applied to the stored scored query.

11. A system according to claim 1, further comprising:

a precomputation engine associating one or more stored documents to the stored query based on at least one of a chosen search document, a set of search documents, regenerated previous search documents, and inverted cached document and query pairings.

12. A method for creating query refinement suggestions, comprising:

matching at least one search document retrieved responsive to a query to one or more stored queries; and

scoring the stored query as a potential query refinement suggestion.

13. A method according to claim 12, further comprising:

matching the at least one search document to one or more stored documents associated with the one or more stored queries.

14. A method according to claim 12, further comprising:

associating a weight with at least one such stored query; and

forming at least one cluster based on the stored query and weight.

15. A method according to claim 14, further comprising:

computing a term vector used in cluster formation from terms extracted from the stored query and based on the weight for the stored query.

16. A method according to claim 15, further comprising:

determining a distance for the term vector; and

forming the at least one cluster relative to the distance.

17. A method according to claim 14, further comprising:

ranking the at least one cluster relative to the at least one other cluster by evaluating a relevance score associated with each search document corresponding to each matched stored document.

18. A method according to claim 17, further comprising:

selecting at least one ranked cluster as a potential refinement cluster.

19. A method according to claim 14, further comprising:

computing a centroid as a weighted center of the at least one cluster; and

computing a score for each stored query relative to the centroid.

20. A method according to claim 19, further comprising:

naming the at least one cluster for at least one scored stored query.

21. A method according to claim 12, further comprising:

applying a threshold to the stored scored query.

22. A method according to claim 12, further comprising:

associating one or more stored documents to the stored query based on at least one of a chosen search document, a set of search documents, regenerated previous search documents, and inverted cached document and query pairings.

23. A computer-readable storage medium holding code for performing the method according to claim 12.

24. An apparatus for creating query refinement suggestions, comprising:

means for matching at least one search document retrieved responsive to a query to one or more stored queries; and

means for scoring the stored query as a potential query refinement suggestion.

25. A system for providing search query refinements, comprising:

an associator associating a stored query and a stored document as a logical pairing and assigning a weight to the logical pairing;

a searcher issuing the search query and producing a set of search documents;

a matcher matching at least one search document to at least one stored document and retrieving the stored query and the assigned weight associated with the matching at least one stored document;

a clusterer forming at least one cluster based on the stored query and the assigned weight associated with the matching at least one stored document; and

a scorer scoring the stored query associated with the matching at least one stored document for the at least one cluster relative to at least one other cluster and suggesting at least one such scored search query as a set of query refinements.

26. A system according to claim 25, further comprising:

a selector selecting one such search document chosen from among the set of search documents responsive to the search query issuance as the at least one such search document.

27. A system according to claim 25, further comprising:

a selector selecting the set of search documents as the at least one such search document.

28. A system according to claim 25, further comprising:

a query log tracking previous search queries; and

a regenerator regenerating a set of previous search documents produced by the previous search queries as the at least one such search document.

29. A system according to claim 25, further comprising:

a cache associating at least one cached document and one or more cached queries as a cached pairing; and

an inverter inverting each cached pairing to associate at least one cached query and one or more cached documents as the at least one such search document.

30. A system according to claim 25, wherein relevancy to the stored query is estimated for the stored document as the weight assigned to the pairing.

31. A system according to claim 30, wherein each such assigned weight for a plurality of pairings corresponding to the stored query and the stored document is summed.

32. A system according to claim 25, wherein each stored query comprises one or more terms, further comprising:

a term vector comprising the terms in the stored query associated with the matching at least one stored document;

a distance determined for the term vector; and

the clusterer forming the at least one cluster relative to the distance.

33. A system according to claim 32, further comprising:

a normalizer normalizing the term vector.

34. A system according to claim 32, further comprising:

an evaluator computing a length of the term vector in multi-dimensional space with each dimension equaling a sum of the weights of the term in a set of associated stored queries.

35. A system according to claim 32, further comprising:

a relevance score assigned to the at least one search document; and

a ranker ranking the at least one cluster relative to the at least one other cluster by the relevance score associated with the matching at least one search document and a number of the matching at least one search document.

36. A system according to claim 35, further comprising:

a selector selecting one of more of the ranked at least one cluster as potential refinement clusters based on the rankings.

37. A system according to claim 36, further comprising:

a centroid computed as a weighted center for each such potential refinement cluster; and

the scorer scoring the stored query associated with the matching at least one stored document for the potential refinement cluster relative to the centroid.

38. A system according to claim 37, further comprising:

an evaluator computing the centroid as a normalized sum of a product of the term vector for each stored query and the relevance score associated with the matching at least one search document.

39. A system according to claim 38, further comprising:

a length of a distance vector determined from the term vector and the centroid; and

the scorer computing the score for the scored query as a product of a number of stored documents with which the stored query is associated and the distance vector length.

40. A system according to claim 25, further comprising:

a selector selecting the stored query associated with the matching at least one stored document relative to a threshold.

41. A system according to claim 25, further comprising:

a sorter sorting the set of query refinements.

42. A system according to claim 25, further comprising:

a presenter presenting the set of query refinements.

43. A system according to claim 25, further comprising:

a set of supplemental query refinements negating each term in the set of query refinements not present in the search query and using the negated terms in combination with the search query as at least one supplemental query refinement.

44. A system according to claim 25, further comprising:

an association database maintaining the pairings.

45. A system according to claim 25, wherein at least one of each such stored document and each such search document is specified as at least one of a Uniform Resource Locator (URL), hyperlink, anchor, and document excerpt.

46. A method for providing search query refinements, comprising:

associating a stored query and a stored document as a logical pairing and assigning a weight to the logical pairing;

issuing the search query and producing a set of search documents;

matching at least one search document to at least one stored document and retrieving the stored query and the assigned weight associated with the matching at least one stored document;

forming at least one cluster based on the stored query and the assigned weight associated with the matching at least one stored document; and

scoring the stored query associated with the matching at least one stored document for the at least one cluster relative to at least one other cluster and suggesting at least one such scored search query as a set of query refinements.

47. A method according to claim 46, further comprising:

selecting one such search document chosen from among the set of search documents responsive to the search query issuance as the at least one such search document.

48. A method according to claim 46, further comprising:

selecting the set of search documents as the at least one such search document.

49. A method according to claim 46, further comprising:

tracking previous search queries; and

regenerating a set of previous search documents produced by the previous search queries as the at least one such search document.

50. A method according to claim 46, further comprising:

associating at least one cached document and one or more cached queries as a cached pairing; and

inverting each cached pairing to associate at least one cached query and one or more cached documents as the at least one such search document.

51. A method according to claim 46, further comprising:

for each such pairing, estimating relevancy to the stored query for the stored document as the weight assigned to the pairing.

52. A method according to claim 51, further comprising:

summing each such assigned weight for a plurality of pairings corresponding to the stored query and the stored document.

53. A method according to claim 46, wherein each stored query comprises one or more terms, further comprising:

computing a term vector comprising the terms in the stored query associated with the matching at least one stored document;

determining a distance determined for the term vector; and

forming the at least one cluster relative to the distance.

54. A method according to claim 53, further comprising:

normalizing the term vector.

55. A method according to claim 53, further comprising:

computing a length of the term vector in multi-dimensional space with each dimension equaling a sum of the weights of the term in a set of associated stored queries.

56. A method according to claim 53, further comprising:

assigning a relevance score to the at least one search document; and

ranking the at least one cluster relative to the at least one other cluster by the relevance score associated with the matching at least one search document and a number of the matching at least one search document.

57. A method according to claim 56, further comprising:

selecting one of more of the ranked at least one cluster as potential refinement clusters based on the rankings.

58. A method according to claim 57, further comprising:

computing a centroid as a weighted center for each such potential refinement cluster; and

scoring the stored query associated with the matching at least one stored document for the potential refinement cluster relative to the centroid.

59. A method according to claim 58, further comprising:

computing the centroid as a normalized sum of a product of the term vector for each stored query and the relevance score associated with the matching at least one search document.

60. A method according to claim 59, further comprising:

determining a length of a distance vector from the term vector and the centroid; and

computing the score for the scored query as a product of a number of stored documents with which the stored query is associated and the distance vector length.

61. A method according to claim 46, further comprising:

selecting the stored query associated with the matching at least one stored document relative to a threshold.

62. A method according to claim 46, further comprising:

sorting the set of query refinements.

63. A method according to claim 46, further comprising:

presenting the set of query refinements.

64. A method according to claim 46, further comprising:

negating each term in the set of query refinements not present in the search query and using the negated terms

in combination with the search query as at least one supplemental query refinement.

65. A method according to claim 46, further comprising:

maintaining the pairings in a database.

66. A method according to claim 46, further comprising:

specifying at least one of each such stored document and each such search document as at least one of a Uniform Resource Locator (URL), hyperlink, anchor, and document excerpt.

67. A computer-readable storage medium holding code for performing the method according to claim 46.

68. An apparatus for providing search query refinements, comprising:

means for associating a stored query and a stored document as a logical pairing and means for assigning a weight to the logical pairing;

means for issuing the search query and means for producing a set of search documents;

means for matching at least one search document to at least one stored document and means for retrieving the stored query and the assigned weight associated with the matching at least one stored document;

means for forming at least one cluster based on the stored query and the assigned weight associated with the matching at least one stored document; and

means for scoring the stored query associated with the matching at least one stored document for the at least one cluster relative to at least one other cluster and means for suggesting at least one such scored search query as a set of query refinements.

69. A system for integrating query refinement candidates, comprising:

a matcher matching at least one search document retrieved responsive to a query to one or more stored documents associated with a stored query and weight and matching at least one further search document retrieved responsive to a candidate query to the one or more stored documents;

a cluster forming at least one cluster based on the stored query and weight associated with each stored document matched responsive to the query and forming at least one further cluster based on the stored query and weight associated with each stored document matched responsive to the candidate query;

a combiner combining the at least one cluster and the at least one further cluster; and

a scorer scoring the stored query for the combined cluster relative to at least one other cluster as a potential query refinement suggestion.

70. A system according to claim 69, further comprising:

a set of candidate query refinements comprising at least one such candidate query.

71. A system according to claim 70, further comprising:

an evaluator assigning at least one such candidate query to the at least one cluster.

72. A system according to claim 71, further comprising:

a builder creating an orthogonal set of candidate query refinements comprising at least one such unassigned query candidate.

73. A method for integrating query refinement candidates, comprising:

matching at least one search document retrieved responsive to a query to one or more stored documents associated with a stored query and weight;

forming at least one cluster based on the stored query and weight associated with each stored document matched responsive to the query;

matching at least one further search document retrieved responsive to a candidate query to the one or more stored documents;

forming at least one further cluster based on the stored query and weight associated with each stored document matched responsive to the candidate query;

combining the at least one cluster and the at least one further cluster; and

scoring the stored query for the combined cluster relative to at least one other cluster as a potential query refinement suggestion.

74. A method according to claim 73, further comprising:

assembling a set of candidate query refinements comprising at least one such candidate query.

75. A method according to claim 74, further comprising:

assigning at least one such candidate query to the at least one cluster.

76. A method according to claim 75, further comprising:

creating an orthogonal set of candidate query refinements comprising at least one such unassigned query candidate.

77. A computer-readable storage medium holding code for performing the method according to claim 73.

78. An apparatus for integrating query refinement candidates, comprising:

means for matching at least one search document retrieved responsive to a query to one or more stored documents associated with a stored query and weight;

means for forming at least one cluster based on the stored query and weight associated with each stored document matched responsive to the query;

means for matching at least one further search document retrieved responsive to a candidate query to the one or more stored documents;

means for forming at least one further cluster based on the stored query and weight associated with each stored document matched responsive to the candidate query;

means for combining the at least one cluster and the at least one further cluster; and

means for scoring the stored query for the combined cluster relative to at least one other cluster as a potential query refinement suggestion.

* * * * *