



US007231399B1

(12) **United States Patent**
Bem et al.

(10) **Patent No.:** **US 7,231,399 B1**
(45) **Date of Patent:** **Jun. 12, 2007**

(54) **RANKING DOCUMENTS BASED ON LARGE DATA SETS**

(75) Inventors: **Jeremy Bem**, Berkeley, CA (US);
Georges R. Harik, Mountain View, CA (US); **Joshua L. Levenberg**, Redwood City, CA (US); **Noam Shazeer**, Stanford, CA (US); **Simon Tong**, Mountain View, CA (US)

(73) Assignee: **Google Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 440 days.

(21) Appl. No.: **10/706,991**

(22) Filed: **Nov. 14, 2003**

(51) **Int. Cl.**
G06F 17/00 (2006.01)
G06F 7/00 (2006.01)

(52) **U.S. Cl.** **707/102; 707/3; 707/5**

(58) **Field of Classification Search** 707/1-7, 707/10, 100, 101, 104.1, 102; 709/218-219, 709/225, 230, 232; 706/12, 25, 47

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,006,222 A	12/1999	Culliss	707/5
6,014,665 A	1/2000	Culliss	707/5
6,078,916 A	6/2000	Culliss	707/5
6,088,692 A *	7/2000	Driscoll	707/5
6,182,068 B1	1/2001	Culliss	707/5
6,285,999 B1 *	9/2001	Page	707/5

6,397,211 B1 *	5/2002	Cooper	707/3
6,463,430 B1 *	10/2002	Brady et al.	707/3
6,539,377 B1	3/2003	Culliss	707/5
6,546,388 B1 *	4/2003	Edlund et al.	707/5
6,546,389 B1 *	4/2003	Agrawal et al.	707/6
6,714,929 B1 *	3/2004	Micaelian et al.	707/4
6,738,764 B2	5/2004	Mao et al.	707/3
6,782,390 B2 *	8/2004	Lee et al.	707/101
6,799,176 B1	9/2004	Page	707/5
7,058,628 B1	6/2006	Page	707/100
2003/0195877 A1 *	10/2003	Ford et al.	707/3
2003/0197837 A1 *	10/2003	Gyu Lee	353/84
2005/0071741 A1	3/2005	Acharya et al.	715/500

OTHER PUBLICATIONS

Justin Boyan et al.; "A Machine Learning Architecture for Optimizing Web Search Engines"; Carnegie Mellon University; May 10, 1996; pp. 1-8.

"Click Popularity—DirectHit Technology Overview"; <http://www.searchengines.com/directhit.html>; Nov. 10, 2003 (print date); 2 pages.

(Continued)

Primary Examiner—Greta Robinson

Assistant Examiner—Jacques Veillard

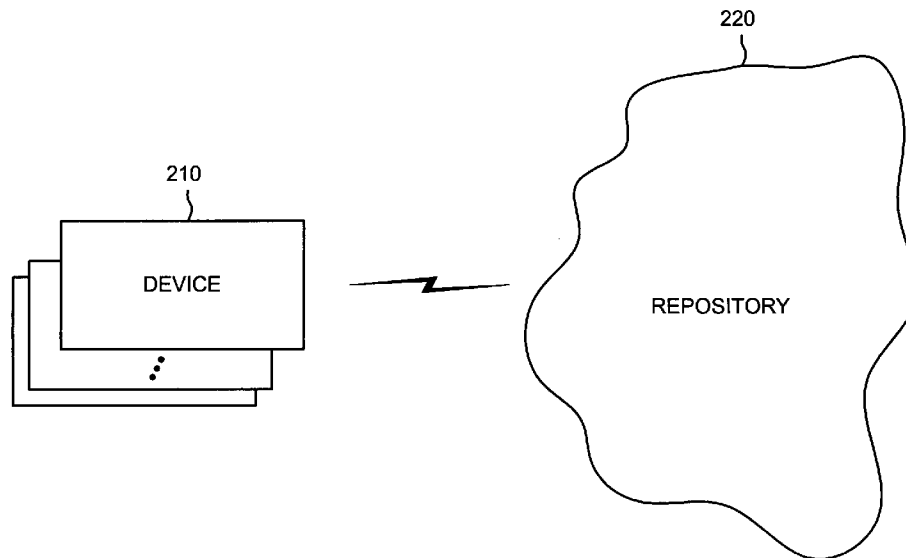
(74) *Attorney, Agent, or Firm*—Harrity Snyder, LLP

(57) **ABSTRACT**

A system ranks documents based, at least in part, on a ranking model. The ranking model may be generated to predict the likelihood that a document will be selected. The system may receive a search query and identify documents relating to the search query. The system may then rank the documents based, at least in part, on the ranking model and form search results for the search query from the ranked documents.

25 Claims, 5 Drawing Sheets

200 →



OTHER PUBLICATIONS

Co-pending U.S. Appl. No. 10/712,263; Jeremy Bem et al.; "Targeting Advertisements Based on Predicted Relevance of the Advertisements"; filed Nov. 14, 2003, 40 pages.

Co-pending U.S. Appl. No. 10/734,584; Jeremy Bem et al.; "Large Scale Machine Learning Systems and Methods"; filed Dec. 15, 2003, 35 pages.

J.H. Friedman, T. Hastie, and R. Tibshirani; "Additive Logistic Regression: a Statistical View of Boosting"; Dept. of Statistics, Stanford University Technical Report; Aug. 20, 1998.

A.Y. Ng and M.I. Jordan; "On Discriminative vs. Generative classifiers: A comparison of logistic regression and naïve Bayes," in

T. Dietterich, S. Becker and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems 14*, Cambridge, MA: MIT Press, 2002.

F. Crestani, M. Lalmas, C. Van Rijsbergen and I. Campbell; "Is This Document Relevant? . . . Probably": A Survey of Probabilistic Models in Information Retrieval"; *ACM Computing Surveys*, vol. 30, No. 4, Dec. 1998.

Jeffrey A. Dean et al., "Ranking Documents Based on User Behavior and/or Feature Data"; U.S. Appl. No. 10/869,057, filed Jun. 17, 2004; 36 pages.

* cited by examiner

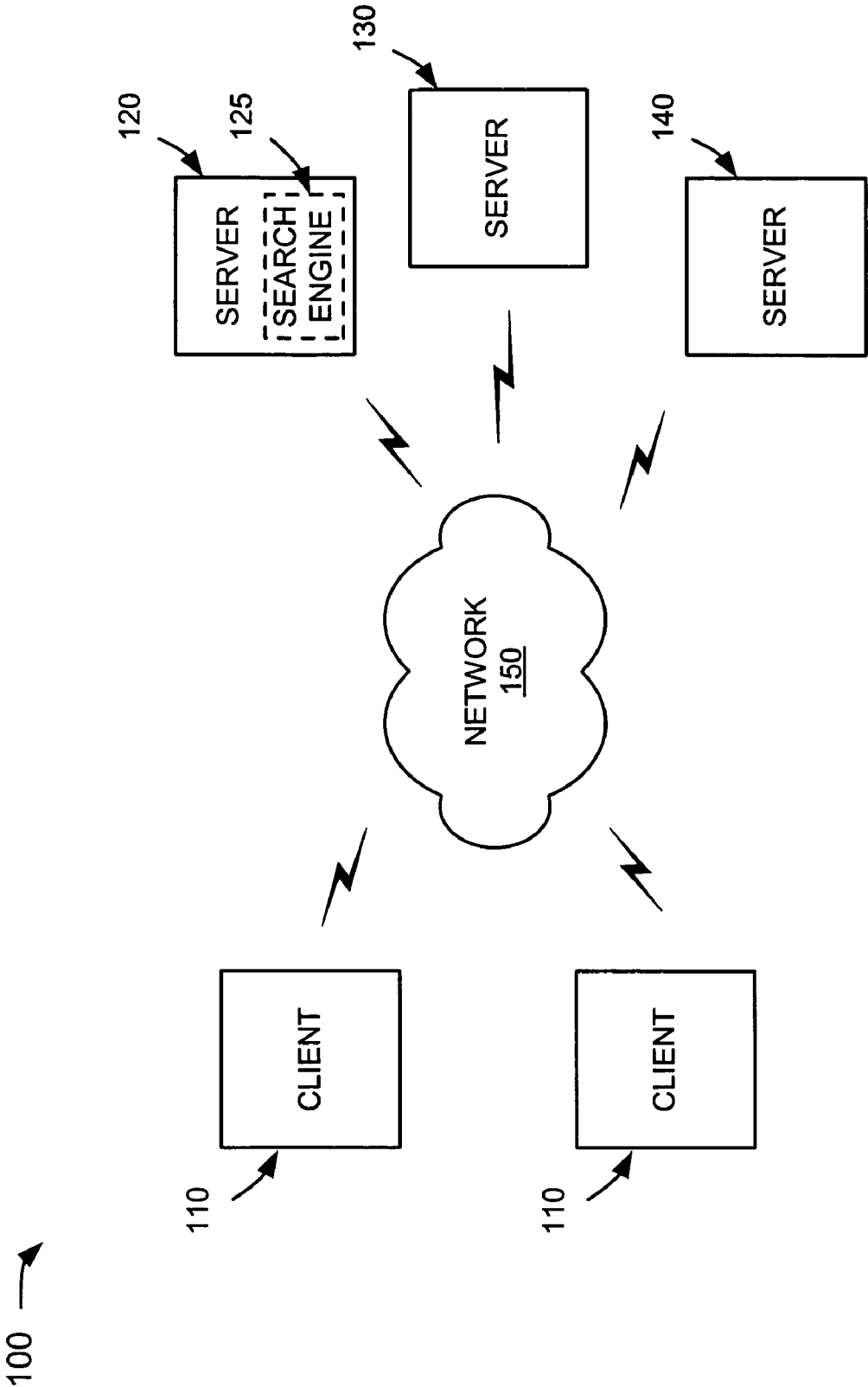


FIG. 1

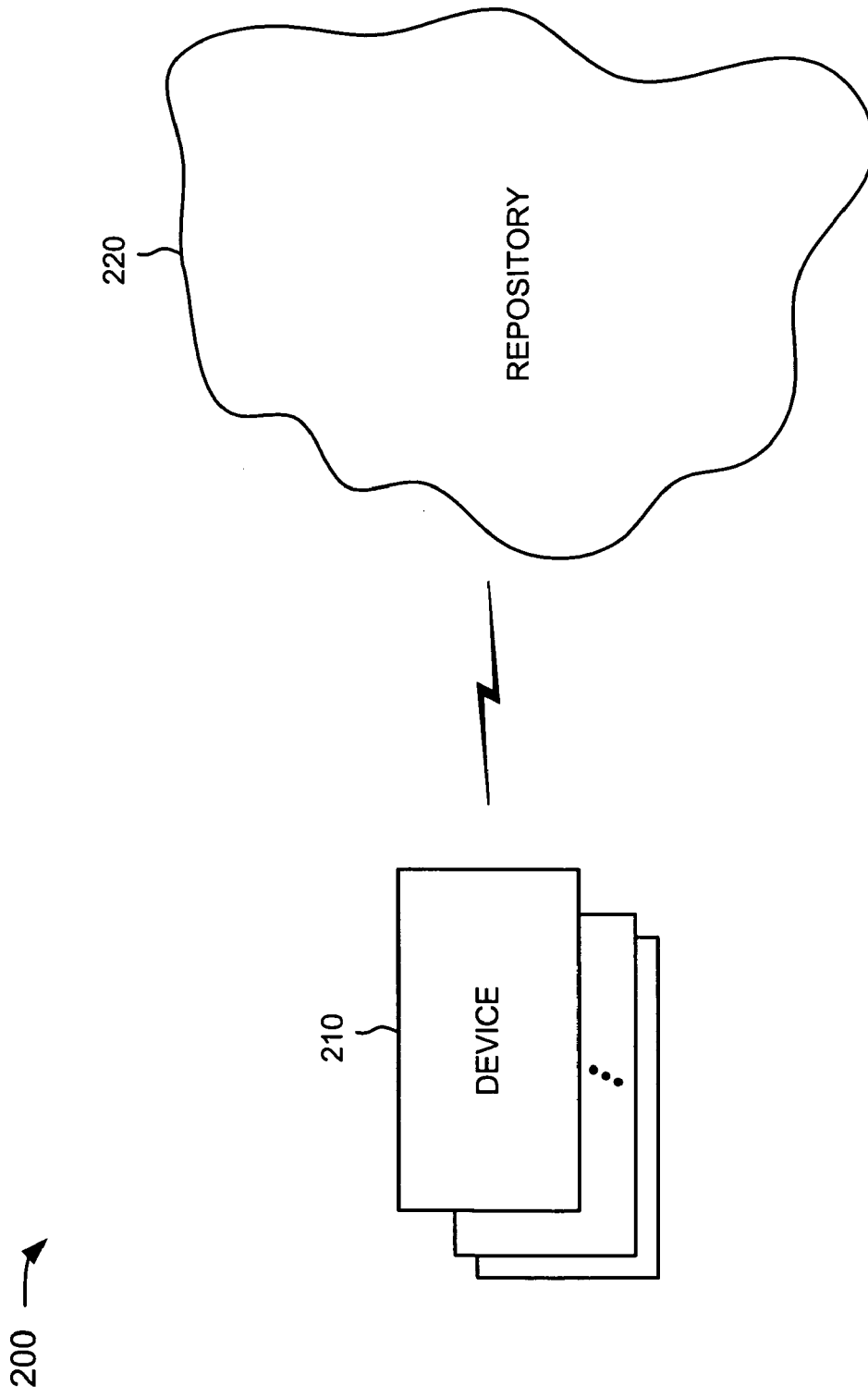


FIG. 2

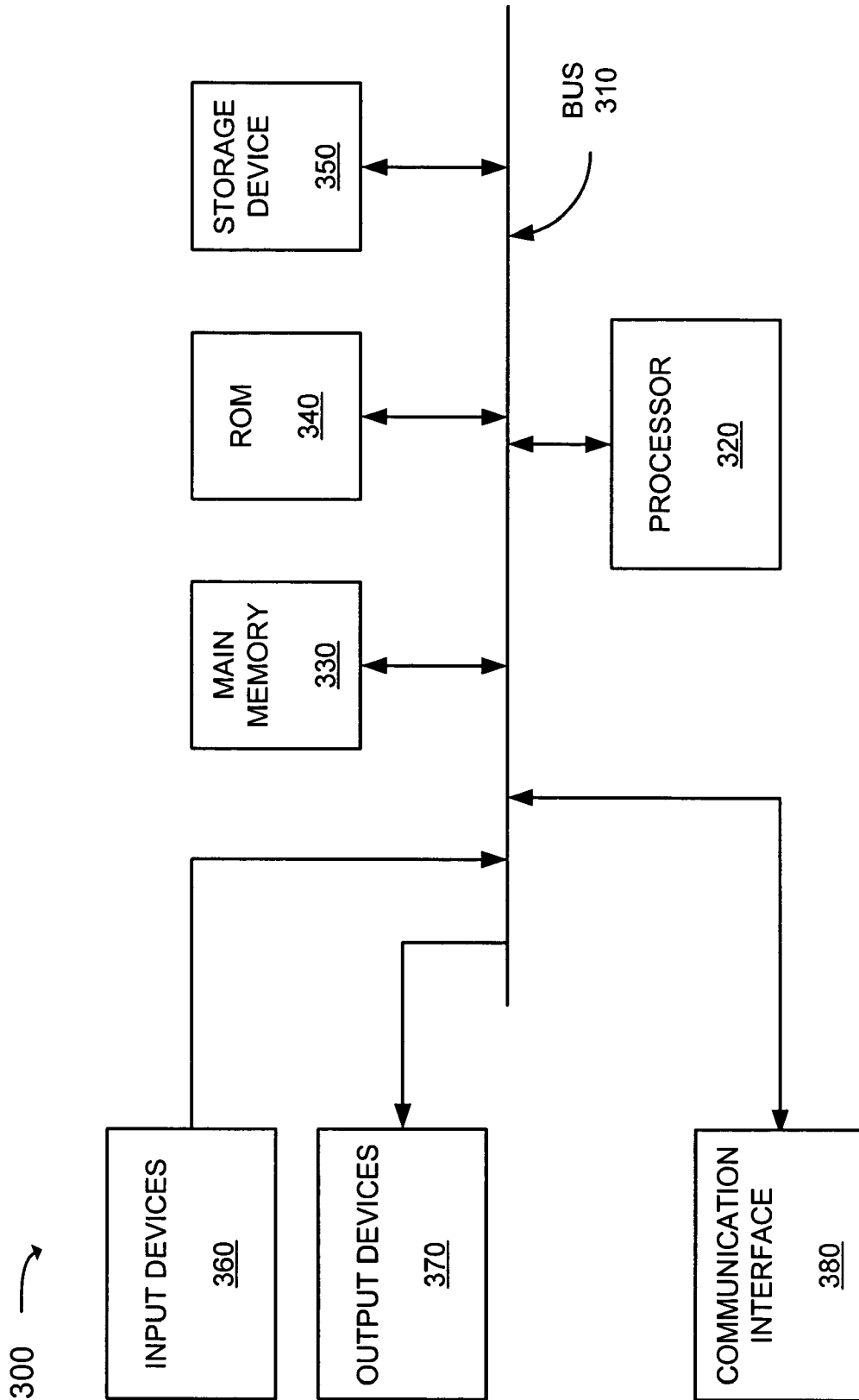


FIG. 3

FIG. 4

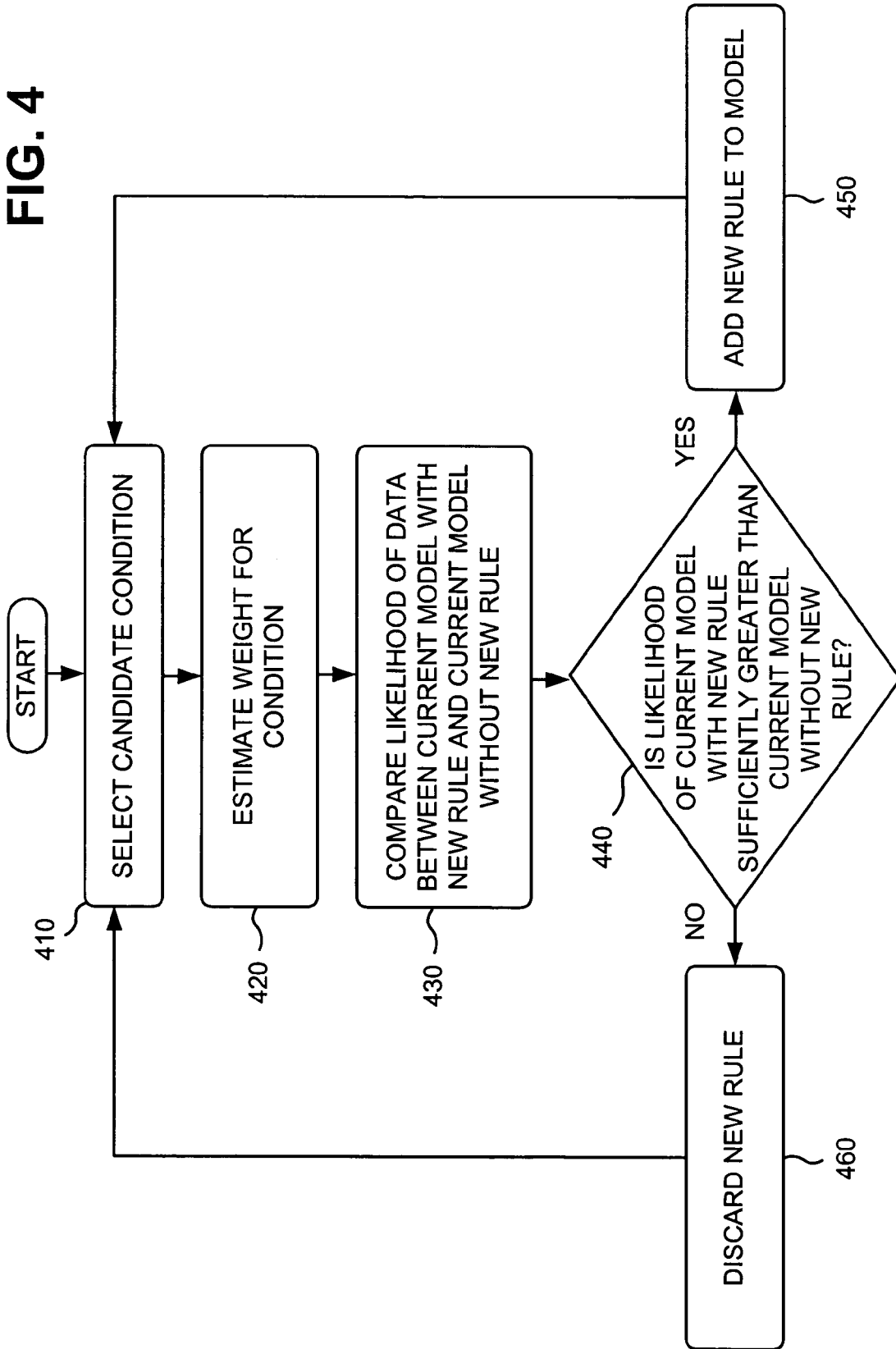
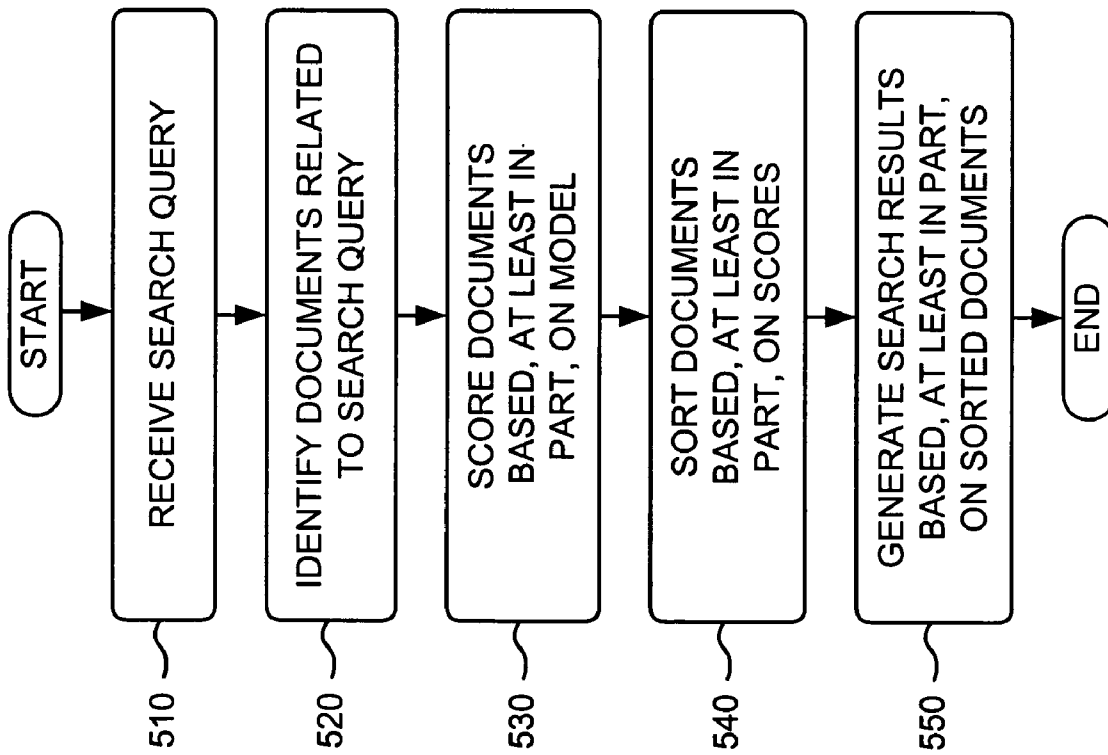


FIG. 5



RANKING DOCUMENTS BASED ON LARGE DATA SETS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to information retrieval and, more particularly, to systems and methods for creating a ranking function from large data sets and using the ranking function to rank documents.

2. Description of Related Art

The World Wide Web ("web") contains a vast amount of information. Locating a desired portion of the information, however, can be challenging. This problem is compounded because the amount of information on the web and the number of new users inexperienced at web searching are growing rapidly.

Search engines attempt to return hyperlinks to web documents in which a user is interested. Generally, search engines base their determination of the user's interest on search terms (called a search query) entered by the user. The goal of the search engine is to provide links to high quality, relevant results to the user based on the search query. Typically, the search engine accomplishes this by matching the terms in the search query to a corpus of pre-stored web documents. Web documents that contain the user's search terms are "hits" and are returned to the user. The search engine oftentimes ranks the documents using a ranking function based on the documents' perceived relevance to the user's search terms. Determining a document's relevance can be a tricky problem.

Accordingly, there is a need for systems and methods that improve the determination of a document's relevance.

SUMMARY OF THE INVENTION

Systems and methods, consistent with the principles of the invention, create a ranking function based, at least in part, on prior information retrieval data, such as query data, user information, and document information, and use the ranking function to rank (or score) documents.

In accordance with one aspect consistent with the principles of the invention, a method for ranking documents is provided. The method may include creating a ranking model that predicts a likelihood that a document will be selected and training the ranking model using a data set that includes tens of millions of instances. The method may also include identifying documents relating to a search query, scoring the documents based, at least in part, on the ranking model, and forming search results for the search query from the scored documents.

According to another aspect, a system for ranking documents is provided. The system may include a repository and a server. The repository may store information corresponding to multiple prior searches. The server may receive a search query from a user, identify documents corresponding to the search query, and rank the identified documents based, at least in part, on a ranking model that includes rules that maximize a likelihood of the repository.

According to yet another aspect, a method for generating a model is provided. The method may include selecting candidate conditions from training data, estimating weights for the candidate conditions, and forming new rules from the candidate conditions and corresponding ones of the weights. The method may also include comparing a likelihood of the training data between a model with the new rules and the

model without the new rules and selectively adding the new rules to the model based, at least in part, on results of the comparing.

According to a further aspect, a method for ranking documents is provided. The method may include receiving a search query, identifying documents relating to the search query, and determining prior probabilities of selecting each of the documents. The method may also include determining a score for each of the documents based, at least in part, on the prior probability of selecting the document and generating search results for the search query from the scored documents.

According to another aspect, a system for generating a model is provided. The system may include a repository and multiple devices. The repository may store training data that includes instances and features, where each of the features corresponds to one or more of the instances. At least one of the devices may create a feature-to-instance index that maps features to instances to which the features correspond, select a candidate condition, request information associated with the candidate condition from other ones of the devices, receive the requested information from the other devices, estimate a weight for the candidate condition based, at least in part, on the requested information, form a new rule from the candidate condition and the weight, and selectively add the new rule to the model.

According to yet another aspect, a system for generating a model is provided. The system includes a repository and multiple devices. The repository may store multiple instances. At least one of the devices may analyze a subset of the instances to identify matching candidate conditions, analyze the candidate conditions to collect statistics regarding predicted probabilities from the matching instances, gather statistics regarding one of the candidate conditions from other ones of the devices, determine a weight associated with the one candidate condition based on at least one of the collected statistics and the gathered statistics, form a rule from the one candidate condition and the weight, and selectively add the rule to the model.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

FIG. 1 is a diagram of an exemplary information retrieval network in which systems and methods consistent with the principles of the invention may be implemented;

FIG. 2 is a diagram of an exemplary model generation system according to an implementation consistent with the principles of the invention;

FIG. 3 is an exemplary diagram of a device according to an implementation consistent with the principles of the invention;

FIG. 4 is a flowchart of exemplary processing for generating a ranking model according to an implementation consistent with the principles of the invention; and

FIG. 5 is a flowchart of exemplary processing for ranking documents according to an implementation consistent with the principles of the invention.

DETAILED DESCRIPTION

The following detailed description of the invention refers to the accompanying drawings. The same reference numbers

in different drawings may identify the same or similar elements. Also, the following detailed description does not limit the invention.

Systems and methods consistent with the principles of the invention may generate a ranking model based, at least in part, on prior information retrieval data, such as data relating to users, queries previously provided by these users, documents retrieved based on these queries, and documents that were selected and not selected in relation to these queries. The systems and methods may use this ranking model as part of a ranking function to rank documents.

Exemplary Information Retrieval Network

FIG. 1 is an exemplary diagram of a network 100 in which systems and methods consistent with the principles of the invention may be implemented. Network 100 may include multiple clients 110 connected to multiple servers 120–140 via a network 150. Network 150 may include a local area network (LAN), a wide area network (WAN), a telephone network, such as the Public Switched Telephone Network (PSTN), an intranet, the Internet, a memory device, another type of network, or a combination of networks. Two clients 110 and three servers 120–140 have been illustrated as connected to network 150 for simplicity. In practice, there may be more or fewer clients and servers. Also, in some instances, a client may perform the functions of a server and a server may perform the functions of a client.

Clients 110 may include client entities. An entity may be defined as a device, such as a wireless telephone, a personal computer, a personal digital assistant (PDA), a lap top, or another type of computation or communication device, a thread or process running on one of these devices, and/or an object executable by one of these device. Servers 120–140 may include server entities that gather, process, search, and/or maintain documents in a manner consistent with the principles of the invention. Clients 110 and servers 120–140 may connect to network 150 via wired, wireless, and/or optical connections.

In an implementation consistent with the principles of the invention, server 120 may optionally include a search engine 125 usable by clients 110. Server 120 may crawl documents (e.g., web pages) and store information associated with these documents in a repository of crawled documents. Servers 130 and 140 may store or maintain documents that may be crawled by server 120. While servers 120–140 are shown as separate entities, it may be possible for one or more of servers 120–140 to perform one or more of the functions of another one or more of servers 120–140. For example, it may be possible that two or more of servers 120–140 are implemented as a single server. It may also be possible that a single one of servers 120–140 is implemented as multiple, possibly distributed, devices.

Exemplary Model Generation System

FIG. 2 is an exemplary diagram of a model generation system 200 consistent with the principles of the invention. System 200 may include one or more devices 210 and a repository 220. Repository 220 may include one or more logical or physical memory devices that may store a large data set (e.g., tens of millions of instances and millions of features) that may be used, as described in more detail below, to create and train a ranking model. The data may include information retrieval data, such as query data, user information, and document information, that may be used to create a model that may be used to rank a particular

document. The query data may include search terms previously provided by users to retrieve documents. The user information may include Internet Protocol (IP) addresses, cookie information, query languages, and/or geographical information associated with the users. The document information may include information relating to the documents presented to the users and the documents that were selected and not selected by the users. In other exemplary implementations, other types of data may alternatively or additionally be stored by repository 220.

Device(s) 210 may include any type of computing device capable of accessing repository 220 via any type of connection mechanism. According to one implementation consistent with the principles of the invention, system 200 may include multiple devices 210. According to another implementation, system 200 may include a single device 210. Device(s) 210 may correspond to one or more of servers 120–140.

FIG. 3 is an exemplary diagram of a device 300 according to an implementation consistent with the principles of the invention. Device 300 may correspond to one or more of clients 110, servers 120–140, and device(s) 210. Device 300 may include a bus 310, a processor 320, a main memory 330, a read only memory (ROM) 340, a storage device 350, one or more input devices 360, one or more output devices 370, and a communication interface 380. Bus 310 may include one or more conductors that permit communication among the components of device 300.

Processor 320 may include any type of conventional processor or microprocessor that interprets and executes instructions. Main memory 330 may include a random access memory (RAM) or another type of dynamic storage device that stores information and instructions for execution by processor 320. ROM 340 may include a conventional ROM device or another type of static storage device that stores static information and instructions for use by processor 320. Storage device 350 may include a magnetic and/or optical recording medium and its corresponding drive.

Input device(s) 360 may include one or more conventional mechanisms that permit an operator to input information to device 300, such as a keyboard, a mouse, a pen, voice recognition and/or biometric mechanisms, etc. Output device(s) 370 may include one or more conventional mechanisms that output information to the operator, including a display, a printer, a speaker, etc. Communication interface 380 may include any transceiver-like mechanism that enables device 300 to communicate with other devices and/or systems.

As will be described in detail below, device 300, consistent with the principles of the invention, may perform certain data-related operations. Device 300 may perform these operations in response to processor 320 executing software instructions contained in a computer-readable medium, such as memory 330. A computer-readable medium may be defined as one or more physical or logical memory devices and/or carrier waves.

The software instructions may be read into memory 330 from another computer-readable medium, such as data storage device 350, or from another device via communication interface 380. The software instructions contained in memory 330 causes processor 320 to perform processes that will be described later. Alternatively, hardwired circuitry may be used in place of or in combination with software instructions to implement processes consistent with the principles of the invention. Thus, implementations consistent with the principles of the invention are not limited to any specific combination of hardware circuitry and software.

Exemplary Model Generation Processing

For purposes of the discussion to follow, the set of data in repository 220 (FIG. 2) may include multiple elements, called instances. It may be possible for repository 220 to store more than 50 million instances. Each instance may include a triple of data: (u, q, d), where u refers to user information, q refers to query data provided by the user, and d refers to document information relating to documents retrieved as a result of the query data and which documents the user selected and did not select.

Several features may be extracted for any given (u, q, d). These features might include one or more of the following: the country in which user u is located, the time of day that user u provided query q, the language of the country in which user u is located, each of the previous three queries that user u provided, the language of query q, the exact string of query q, the word(s) in query q, the number of words in query q, each of the words in document d, each of the words in the Uniform Resource Locator (URL) of document d, the top level domain in the URL of document d, each of the prefixes of the URL of document d, each of the words in the title of document d, each of the words in the links pointing to document d, each of the words in the title of the documents shown above and below document d for query q, the number of times a word in query q matches a word in document d, the number of times user u has previously accessed document d, and other information. In one implementation, repository 220 may store more than 5 million distinct features.

To facilitate fast identification of correspondence between features and instances, a feature-to-instance index may be generated that links features to the instances in which they are included. For example, for a given feature f, the set of instances that contain that feature may be listed. The list of instances for a feature f is called the "hitlist for feature f." Thereafter, given a set of features f_0, \dots, f_n , the set of instances that contain those features can be determined by intersecting the hitlist for each of the features f_0, \dots, f_n .

Other information may also be determined for a given (u, q, d). This information may include, for example, the position that document d was provided within search results presented to user u for query q, the number of documents above document d that were selected by user u for query q, and a score ("old score") that was assigned to document d for query q. The old score may have been assigned by search engine 125 or by another search engine.

A ranking model may be created from this data. The model uses the data in repository 220 as a way of evaluating how good the model is. The model may include rules that maximize the log likelihood of the data in repository 220. The general idea of the model is that given a new (u, q, d), the model may predict whether user u will select a particular document d for query q. As will be described in more detail below, this information may be used to rank document d for query q and user u.

FIG. 4 is a flowchart of exemplary processing for generating a ranking model according to an implementation consistent with the principles of the invention. This processing may be performed by a single device 210 or a combination of multiple devices 210.

To facilitate generation of the ranking model, a prior probability of selection may be determined. As described above, the following information may be determined for any given instance (u, q, d): the position that document d occupied within the search results presented to user u for query q; the old score that was assigned to document d for

query q; and the number of documents above document d that were selected by user u for query q. Based on this information, a function may be created that maps from these three values to a probability of selection:

$$P(\text{select}|\text{position, old score, number of selections above}).$$

This "prior" probability of selection may provide the initial probability of document selection without considering any of the features. It uses the position, the old score, and the number of selections of documents above this document.

A set of instances based on the same or a different set of instances may be used as "training data" D. For each instance (u, q, d) in the training data D, its features (f_0, f_1, \dots, f_n) may be extracted. For example, f_0 may be the feature corresponding to "the word 'tree' appears in the query." In this implementation, the feature f_0 may include a boolean value, such that if "tree" appears in query q then the value of f_0 is one, otherwise the value of f_0 is zero. In other implementations, the features may include discrete values. It may be assumed that many of the features will have values of zero. Accordingly, a sparse representation for the features of each instance may be used. In this case, each instance may store only features that have non-zero values.

Therefore, for each instance (u, q, d), the following information is available: its set of features, whether document d was selected by user u for query q, and its prior probability of selection. A "condition" C is a conjunction of features and possibly their complements. For example, a condition that includes two features is: "tree" is in query q and the domain of document d is "trees.com," and a condition that includes a feature and a complement of a feature is: "football" is in query q and the user did not provide the query from "www.google.co.uk." For any instance (u, q, d), the value of its features may determine the set of conditions C that apply. A "rule" is a condition C and a weight w, represented as (C, w). The ranking model M may include a set of rules and a prior probability of selection. To generate the model M, the set of conditions C_1, \dots, C_n and the values of the weights w_1, \dots, w_n need to be determined.

Based on this information, a function may be created that maps conditions to a probability of selection:

$$P(\text{select}|C_1, \dots, C_n, \text{position, old score, number of selections above}).$$

The posterior probability of a selection given a set of conditions, $P(\text{select}|C_1, \dots, C_n, \text{position, old score, number of selections above})$, may be determined using the function:

$$\text{Log} \{P(\text{select=false}|C_1, \dots, C_n, \text{position, old score, number of selections above})\}$$

$$P(\text{select=true}|C_1, \dots, C_n, \text{position, old score, number of selections above})\}$$

$$= \text{Sum}_i \{ -w_i I(C_i) \} + \text{Log} \{ P(\text{select=false}|\text{position, old score, number of selections above}) / P(\text{select=true}|\text{position, old score, number of selections above}) \},$$

where $I(C_i) = 0$ if $C_i = \text{false}$, and $I(C_i) = 1$ if $C_i = \text{true}$.

To generate the model M, processing may start with an empty model M that includes the prior probability of selection. A candidate condition C may be selected (act 410). In one implementation, candidate conditions may be selected from the training data D. For example, for each instance in the training data D, combinations of features that are present in that instance (or complements of these features) may be

chosen as candidate conditions. In another implementation, random sets of conditions may be selected as candidate conditions. In yet another implementation, single feature conditions may be considered for candidate conditions. In a further implementation, existing conditions in the model M may be augmented by adding extra features and these augmented conditions may be considered as candidate conditions. In yet other implementations, candidate conditions may be selected in other ways.

A weight w for condition C may then be estimated (act 420). The weight w may be estimated by attempting to maximize the log likelihood of the training data D given the model M augmented with rule (C, w) —that is, find the weight that maximizes $\text{Log } P(\text{DIM}, (C, w))$, where “ $M, (C, w)$ ” denotes the model M with rule (C, w) added if condition C is not already part of the model M , and w is the weight for condition C .

The likelihood of the training data D may be compared between the current model with the new rule (C, w) and the current model without the new rule (i.e., $P(\text{DIM}, (C, w))$ vs. $P(\text{DIM})$) (acts 430 and 440). If $P(\text{DIM}, (C, w))$ is sufficiently greater than $P(\text{DIM})$, then the new rule (C, w) is added to the model M (act 450). A penalty or “Cost” for each condition C may be used to aid in the determination of whether $P(\text{DIM}, (C, w))$ is sufficiently greater than $P(\text{DIM})$. For example, if condition C includes many features, or if the features of condition C are quite rare (e.g., “does the word ‘mahogany’ appear in the query”), then the cost of condition C could be high. The rule (C, w) may then be added to the model M if:

$$\text{Log } \{P(\text{DIM}, (C, w))\} - \text{Log } \{P(\text{DIM})\} > \text{Cost}(C).$$

If $P(\text{DIM}, (C, w))$ is not sufficiently greater than $P(\text{DIM})$, then the new rule (C, w) is discarded (i.e., not added to the model M), possibly by changing its weight to zero (act 460). In either event, processing may then return to act 410, where the next candidate condition is selected. Processing may continue for a predetermined number of iterations or until all candidate conditions have been considered.

Exemplary Process for Ranking Documents

FIG. 5 is a flowchart of exemplary processing for ranking documents according to an implementation consistent with the principles of the invention. Processing may begin with a user providing one or more search terms as a search query for searching a document corpus. In one implementation, the document corpus is the Internet and the vehicle for searching this corpus is a search engine, such as search engine 125 (FIG. 1). The user may provide the search query to search engine 125 via web browser software on a client, such as client 110 (FIG. 1).

Search engine 125 may receive the search query and act upon it to identify documents (e.g., web pages) related to the search query (acts 510 and 520). A number of techniques exist for identifying documents related to a search query. One such technique might include identifying documents that contain the one or more search terms as a phrase. Another technique might include identifying documents that contain the one or more search terms, but not necessarily together. Other techniques might include identifying documents that contain less than all of the one or more search terms, or synonyms of the one or more search terms. Yet other techniques are known to those skilled in the art.

Search engine 125 may then score the documents based on the ranking model described above (act 530). With regard to each document, search engine 125 may identify a new

instance (u, q, d) that corresponds to this user search, where u refers to the user, q refers to the search query provided by the user, and d refers to the document under consideration. Search engine 125 may extract the features from the new instance and determine which rules of the ranking model apply. Search engine 125 may then combine the weight of each rule with the prior probability of selection for (u, q, d) to determine the final posterior probability of the user u selecting this document d for query q . Search engine 125 may use the final posterior probability as the score for the document. Alternatively, search engine 125 might use the final posterior probability as one of multiple factors in determining the score of the document.

Search engine 125 may sort the documents based on their scores (act 540). Search engine 125 may then formulate search results based on the sorted documents (act 550). In an implementation consistent with the principles of the invention, the search results may include references to the documents, such as links to the documents and possibly a textual description of the links. In another implementation, the search results may include the documents themselves. In yet other implementations, the search results may take other forms.

Search engine 125 may provide the search results as a HyperText Markup Language (HTML) document, similar to search results provided by conventional search engines. Alternatively, search engine 125 may provide the search results according to a protocol agreed upon by search engine 125 and client 110 (e.g., Extensible Markup Language (XML)).

Search engine 125 may further provide information concerning the user, the query provided by the user, and the documents provided to the user to help improve the ranking model. For example, server 120 may store this information in repository 220 or provide it to one of devices 210 to be used as training data for training the model.

Even Larger Data Sets

When the data set within repository 220 becomes very large (e.g., substantially more than a few million instances), multiple devices 210 may be configured as a distributed system. For example, devices 210 may be capable of communicating with each other and with repository 220, as illustrated in FIG. 2.

According to one exemplary implementation of the distributed system, each device 210 may be responsible for a subset of the instances within repository 220. Each device 210 may possibly store its subset of instances in local memory. Each device 210 may also build its own feature-to-instance index for its subset of instances. As described previously, the feature-to-instance index may facilitate fast identification of correspondence between features and instances by linking features to the instances in which they are included.

One or more devices 210 (or possibly all) may identify candidate conditions to be tested. Suppose that a device 210 “DV” has a candidate condition C that it wants to test (e.g., for which it wants to determine a weight). Device DV may send out a “stats” request to other devices 210 asking for statistics for condition C . For example, device DV may broadcast the stats request that includes (or identifies) the condition C for which device DV desires statistics.

The other devices 210 may receive condition C and use their own feature-to-instance index to find their instances that satisfy condition C . For example, the receiving devices 210 may identify the features making up condition C .

Assume, for example, that condition C includes a combination of features f_i and f_j . The receiving devices **210** may then determine the set of instances that contain those features by intersecting the hitlist for features f_i and f_j .

The receiving device **210** may then compute statistics about predicted probabilities for those instances. Each receiving device **210** may then send the statistics for the set of matching instances back to device DV. Device DV may use the statistics to estimate a weight for condition C. Device DV may, as described above, determine whether to add the rule containing condition C to the model. If device DV decides to add the rule containing condition C to the model, it may send an “update” request to all of the other devices **210** informing them of the new rule (i.e., the new condition and its weight).

According to another exemplary implementation of the distributed system, multiple devices **210** may be used to further increase throughput and capacity. That is, this implementation can handle more total instances and test more conditions per second. It does this by first generating a collection of conditions to test, and then optimizing them in a batch. It is desirable, however, to be able to update the model everywhere immediately when a new rule is added since correlated conditions cannot be considered in isolation without introducing unacceptable weight oscillations that prevent convergence. Therefore, implementations consistent with the principles of the invention may rapidly feed back the effects of accepted rules into the statistics used to decide the weight of future rules.

As before, the model generation process may be divided into iterations. Rules may be tested or have their weights optimized once per iteration. Each iteration may be broken into two phases: a candidate rule generation phase and a rule testing and optimization phase. The rule testing and optimization phase may determine the weights for conditions generated in the candidate rule generation phase, and accept rules into the model if their benefit (difference in log likelihood) exceeds their cost.

As described previously, there are several possible ways of generating candidate conditions in the candidate rule generation phase. For example, candidate conditions might include all conditions with one feature, all conditions with two features that co-occur in some instance, and all extensions of existing rules by one feature (where the combination is in some instance). It may be beneficial to consider only those conditions that match some minimum number of instances. There are a couple of strategies for accomplishing this: identify conditions that appear multiple times in some fraction of the instances (divided among all of devices **210** and then summed), or identify conditions that appear multiple times on one device **210** and then gather the results together to remove duplicates. As a further optimization, extensions of only those rules added in the last iteration may be determined and added to the candidate rules generated in previous iterations.

The conditions to be tested may be distributed to every device **210**. Each device **210** may analyze its share of the instances to identify candidate conditions that match each instance. For example, devices **210** may determine matching conditions for each instance by looking up the features in the instance in a tree data structure. Devices **210** may record information concerning the matching conditions and instances as (condition, instance number) pairs. Each device **210** may sort the (condition, instance number) pairs and use them to iterate through the conditions to collect statistics about predicted probabilities from the matching instances.

For each condition, devices **210** may send the statistics associated with the condition to a device **210** designated to handle that condition (e.g., a device **210** selected, for instance, based on a hash of the condition). When all of the statistics for a given condition have been aggregated on a single device **210**, that device **210** may determine the optimal weight of the rule and determine whether the rule should be added to the model M, as described above. The result may be broadcast to all devices **210** (or to those devices that sent statistics). The output of this phase is new weights for all existing rules (possibly zero if the rule is discarded) and a list of new rules.

CONCLUSION

Systems and methods consistent with the principles of the invention may rank search results based on a ranking model that may be generated based, at least in part, on prior information retrieval data, such as data relating to users, queries previously provided by these users, documents retrieved based on these queries, and which of these documents were selected and not selected in relation to these queries.

The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For example, while series of acts have been described with regard to FIGS. 4 and 5, the order of the acts may be modified in other implementations consistent with the principles of the invention. Also, non-dependent acts may be performed in parallel. Further, the acts may be modified in other ways. For example, in another exemplary implementation, acts 420–430 of FIG. 4 may be performed in a loop for a number of iterations to settle on a good weight. In the context of multiple devices **210**, this may mean that device DV sends out multiple stats requests for a particular condition.

It will also be apparent to one of ordinary skill in the art that aspects of the invention, as described above, may be implemented in many different forms of software, firmware, and hardware in the implementations illustrated in the figures. The actual software code or specialized control hardware used to implement aspects consistent with the present invention is not limiting of the present invention. Thus, the operation and behavior of the aspects were described without reference to the specific software code—it being understood that one of ordinary skill in the art would be able to design software and control hardware to implement the aspects based on the description herein.

What is claimed is:

1. A computer implemented method, comprising:
 - creating a ranking model that predicts a likelihood that a document will be selected by:
 - storing information associated with a plurality of prior searches,
 - determining a prior probability of selection based, at least in part, on the information associated with the prior searches, and
 - generating the ranking model based, at least in part on the prior probability of selection;
 - training the ranking model using a data set that includes approximately tens of millions of instances;
 - identifying documents relating to a search query;
 - scoring the documents based, at least in part, on the ranking model;

11

forming search results for the search query from the scored documents; and outputting the search results.

2. The method of claim 1, wherein the information associated with the prior searches includes, for each of a plurality of documents associated with the prior searches, at least one of a position occupied by the document within prior search results, a score assigned to the document, or a number of documents listed above the document in the prior search results that were selected.

3. The method of claim 1, wherein the creating a ranking model includes:

storing training data,
extracting features from the training data, and
generating conditions that include one or more of the extracted features.

4. The method of claim 3, wherein the creating a ranking model further includes:

selecting one of the conditions as a candidate condition,
estimating a weight for the candidate condition,
forming a new rule from the candidate condition and the estimated weight,
comparing a likelihood of the training data between a current model with the new rule and the current model without the new rule, and
selectively adding the new rule to the current model based, at least in part, on a result of the comparison.

5. The method of claim 4, wherein the selecting one of the conditions as a candidate condition includes at least one of: creating the candidate condition from combinations of features or complements of features in the training data, randomly selecting one of the conditions as the candidate condition,
selecting one of the conditions that includes a single one of the features as the candidate condition, or
augmenting one of the conditions by adding one or more features to the one condition to form the candidate condition.

6. The method of claim 4, wherein the estimating a weight includes determining a weight that maximizes a likelihood of the training data given the model.

7. The method of claim 4, wherein the selectively adding the new rule to the current model includes adding the new rule to the current model when a likelihood of the training data occurring when the current model includes the new rule is greater than when the current model does not include the new rule.

8. The method of claim 4, wherein the selectively adding the new rule to the current model further includes:

associating a cost with each of the conditions, and
determining whether to add the new rule to the current model based, at least in part, on the cost associated with the candidate condition.

9. The method of claim 4, further comprising:
performing a number of iterations including estimating the weight, forming the new rule, and comparing the likelihood of the training data.

10. The method of claim 1, wherein the data set also includes approximately millions of features.

11. The method of claim 1, wherein the scoring the documents includes:

forming an instance that corresponds to the search query and one of the documents,
extracting features associated with the instance,
identifying rules in the ranking model that apply based, at least in part, on the extracted features, each of the identified rules including a weight, and

12

combining the weights of the identified rules with a prior probability of selection corresponding to the instance to generate a score for the one document.

12. The method of claim 11, wherein the instance includes user information corresponding to a user who provided the search query, query data corresponding to the search query, and document information corresponding to the one document.

13. The method of claim 1, wherein the scoring the documents includes:

determining a prior probability of selection corresponding to the search query and one of the documents, and
generating a score for the one document based, at least in part, on the determined prior probability of selection.

14. The method of claim 13, wherein the generating a score for the one document includes using the determined prior probability of selection as one of a plurality of factors in determining the score for the one document.

15. A system implemented within one or more computer devices, comprising:

means for receiving a search query;
means for identifying documents relating to the search query;

means for ranking the documents based, at least in part, on a ranking model trained on a large data set that includes approximately millions of features, the means for ranking includes:

means for determining a prior probability of selection corresponding to the search query and one of the identified documents, and

means for determining a rank for the one document based, at least in part, on the determined prior probability of selection;

means for forming search results for the search query from the ranked documents; and

means for outputting the search results.

16. A system, comprising:

a repository configured to store information corresponding to a plurality of prior searches; and

a server configured to:

receive a search query from a user,
identify documents corresponding to the search query,
rank the identified documents based, at least in part, on a ranking model that includes rules that maximize a likelihood of the repository, when ranking the identified documents, the server is configured to:

determine a prior probability of selection corresponding to the search query and one of the identified documents, and

determine a rank for the one document based, at least in part, on the determined prior probability of selection, and

output the ranked documents.

17. A system, comprising:

a repository configured to store information corresponding to a plurality of prior searches; and

a server configured to:

receive a search query from a user,
identify documents corresponding to the search query,
rank the identified documents based, at least in part, on a ranking model that includes rules that maximize a likelihood of the repository, when ranking the identified documents, the server is configured to:

determine a prior probability of selection corresponding to the search query and one of the identified documents, and

13

determine a rank for the one document based, at least in part, on the determined prior probability of selection, and

output the ranked documents.

18. The system of claim 17, wherein the repository is further configured to store a plurality of features associated with the instances.

19. The system of claim 18, wherein when ranking the documents, the server is configured to:

identify one of the instances that corresponds to the search query and one of the identified documents,

determine features associated with the identified instance, identify rules in the ranking model that apply based, at least in part, on the determined features, each of the identified rules including a weight, and

combine the weights of the identified rules with a prior probability of selection corresponding to the identified instance to determine a rank for the one document.

20. The system of claim 19, wherein the user information of the identified instance includes information corresponding to the user who provided the search query, the query data of the identified instance includes information corresponding to the search query, and the document information of the identified instance includes information corresponding to the one document.

21. The system of claim 16 wherein when determining a rank for the one document, the server is configured to use the determined prior probability of selection as one of a plurality of factors in determining the rank for the one document.

22. The system of claim 16, wherein the repository stores approximately tens of millions of instances and approximately millions of features associated with the plurality of prior searches.

23. A method, comprising:

receiving a search query;

identifying documents relating to the search query;

determining prior probabilities of selecting each of the documents, where the prior probability of selecting one of the documents is determined based, at least in part, on data regarding at least one of a position of the document within search results, a prior score assigned

14

to the document, or a number of documents above the document in the search results that were selected;

determining a score for each of the documents based, at least in part, on the prior probability of selecting the document;

generating search results for the search query from the scored documents, and

outputting the search results.

24. A method, comprising:

creating a ranking model that predicts a likelihood that a document will be selected by:

storing information associated with a plurality of prior searches,

determining a prior probability of selection based, at least in part, on the information associated with the prior searches, and

generating the ranking model based, at least in part, on the prior probability of selection;

identifying documents relating to a search query;

scoring the documents based, at least in part, on the ranking model;

forming search results for the search query from the scored documents; and

outputting the search results.

25. A method, comprising:

receiving a search query;

identifying documents relating to the search query;

determining a prior probability of selecting one of the documents, the prior probability of selecting the one document is determined based, at least in part, on data regarding at least one of a position of the one document within search results, a prior score assigned to the one document, or a number of documents above the one document in the search results that were selected;

determining a score for the one document based, at least in part, on the prior probability of selecting the one document;

generating a list of search results that includes the one document based on the determined score; and

outputting the list of search results.

* * * * *