(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 May 2006 (04.05.2006)

PCT

(10) International Publication Number
WO 2006/047407 A2

(51) International Patent Classification:
G06F 12/00 (2006.01)

(21) International Application Number:
PCT/US2005/038167

(22) International Filing Date: 20 October 2005 (20.10.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/973,514 26 October 2004 (26.10.2004) US

(71) Applicant *(for all designated States except US)*: **YAHOO! INC** [US/US]; 701 Frist Avenue, Sunnyvale, CA 94089 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **LEU, Eric** [US/US]; 5833 Chambertin Drive, San Jose, CA 95118 (US).

(74) Agents: **SALTZBERG, Robert, A.** et al.; Morrison & Foerster LLP, 425 Market Street, San Francisco, CA 94105-2482 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

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

WO 2006/047407 A2

(54) Title: METHOD OF INDEXING GATEGORIES FOR EFFICIENT SEARCHING AND RANKING

(57) **Abstract:** A document management system indexes categories for efficient retrieval based on a category sequence. The category sequence is generated such that for any first, second and third categories appearing in the sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories. A category index implemented in this manner significantly reduces the number of similarity distance computations that are performed when searching for categories and documents that are most relevant to content presented on a web page.

METHOD OF INDEXING CATEGORIES FOR EFFICIENT SEARCHING AND RANKING

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates generally to document management, and more particularly, to a method and system for indexing document categories for efficient searching and ranking.

Description of the Related Art

[0002] Banner ads are generally viewed as an ineffective form of advertising on the Internet. As a way to increase the effectiveness of banner ads, some web site operators have employed content matching. With content matching, banner ads are not randomly selected for display on a web page, but are "matched" to the content of the web page.

[0003] One way to find "matching" banner ads is to do an exhaustive search of banner ads in the ad inventory to determine which banner ads are the most relevant to the web page content, but this method is too inefficient when the ad inventory is large. Another way to find "matching" banner ads is to randomly pick a small set of banner ads from the ad inventory and then select those banner ads that are most relevant to the web page content. The drawback of this method is that there is no assurance that the banner ads initially selected are the most relevant or even relevant at all.

[0004] A better way to find "matching" banner ads is to classify the banner ads into categories and select banner ads from those categories that are most relevant to the web page content. This method, however, requires that the relevance of a category to the web page content be computed for each category. As a result, when the number of categories is large, this method becomes computationally too expensive.

SUMMARY OF THE INVENTION

[0005] The invention provides a category indexing method and system that significantly reduces the number of computations that are performed when searching for categories that are most relevant to content presented on a web page. As an example application of the invention, highly relevant online ads can be matched to the web page content very rapidly so that the matching online ads can be retrieved and transmitted with the web page content on a real-time basis.

[0006] According to an embodiment of the invention, categories are indexed for efficient retrieval based on a category sequence. The category sequence is generated such that, for any first, second and third categories appearing in the sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories.

[0007] The category index may be used to find online ads or other documents that are most relevant to content to be transmitted for display at a user computer. The most relevant online ads or documents are obtained by: (i) identifying and ranking the categories that are most relevant to the category of the content to be transmitted; and (ii) retrieving the online ads or documents from the categories beginning with the highest ranked category until a desired number of documents are collected.

[0008] The document management system according to an embodiment of the invention includes a category database that has a plurality of category records and a processor programmed to generate a set of categories most relevant to a reference category using the category database. Each category record in the category database includes an identifier for the category and a position of the category in a category sequence, wherein, for any first, second and third categories appearing in the category sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] So that the manner in which the above recited features of the present invention can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0010] FIG. 1 illustrates a block diagram of a document management system that implements an embodiment of the invention;

[0011] FIG. 2 is a sample category tree;

[0012] FIG. 3 is a flow diagram illustrating the process steps for generating a valid category sequence;

[0013] FIG. 4 is a flow diagram illustrating the process steps for finding the most relevant category;

[0014] FIG. 5 is a sample category tree showing an example in which qualifying categories do not include all categories;

[0015] FIG. 6 is a flow diagram illustrating the process steps for finding the N most relevant categories;

[0016] FIG. 7 is a flow diagram illustrating the process steps for finding M documents that are most relevant with respect to a given category;

[0017] FIG. 8 is a sample category tree where some category nodes have more than one parent node;

[0018] FIG. 9 is a sample category tree showing pivot category nodes; and

[0019] FIG. 10 is a flow diagram illustrating the process steps for finding the most relevant category for a category that has pivot category nodes as ancestor nodes.

DETAILED DESCRIPTION

[0020] FIG. 1 illustrates a block diagram of a document management system 10 that implements an embodiment of the invention. The document management system 10 includes a web server 12 that receives HTTP requests made over the Internet 13 by a remote computer 14 and transmits web page content and relevant banner ads in response to the HTTP requests for display at the remote computer 14.

[0021] The document management system 10 further includes a content database 15, a category database 16, and an ad server 17. The content database 15 stores content, including online ads, news articles and other content. As each such item is stored in the content database 15, it is classified into one or more categories. The category database 16 stores metadata information about the categories, including weight values of categories, category hierarchy, and indexes. The ad server 17 is programmed to select banner ads that are most relevant to a given category and pass them to the web server 12.

[0022] The categories are related to one another in a hierarchical manner. FIG. 2 is a node tree for a sample group of categories. Categories A, B, C, D, E, F and G are illustrated in FIG. 2 as nodes of the node tree (also referred to as a category tree). A is the root node. B and C are child nodes of A. D, E and F are child nodes of B. G is a child node of category D.

[0023] By way of definition, a "parent" node of X is a node directly from which X descends. As an example, A is a parent node of both B and C. An "ancestor" node of X is any node from which X descends. As an example, A, B and D are ancestor nodes of G. A descendent node of X is any node which descends from X. As an example, D, E, F and G are descendent nodes of B. A "leaf" node is a node that has no child nodes. As an example, C, E, F and G are leaf nodes.

[0024] In the embodiment of the invention illustrated herein, each node in FIG. 2 is assigned a weight value. The weight value of each node is less than or equal to the weight value of its parent node. The weight values of nodes A – G are:

Node	A	B	C	D	E	F	G
Weight	20	6	1	2	1	2	1

[0025] The weight value of a node provides a rough indication of the number of nodes that descend from it. A node with a higher weight value generally has more descendent nodes than a node with a lower weight value. As a special example, the weight value of each node may be assigned 1, 2 or 3 based on the count of its descendant nodes. If the count is in the range of 0 and 15, the weight value of 1 is assigned. If the count is in the range of 16 and 80, the weight value of 2 is assigned. If the count is over 80, the weight value of 3 is assigned.

[0026] The weight values are used to compute a similarity distance between the nodes. In the example illustrated herein, the following similarity distance function is used. For any two nodes X and Y, their similarity distance, indicated as Similarity(X, Y), is the weight value of the lowest common ancestor node of X and Y. The similarity distance between node X and itself is the weight value of node X. As an example, in FIG. 2, Similarity(A, C) = 20; Similarity(D, F) = 6, and Similarity(A, A) = 20. A larger number indicates that the two nodes are less similar. The similarity function is also symmetric, i.e., Similarity(X, Y) = Similarity(Y, X). The weight values of the nodes can be adjusted to tune the similarity distance between the nodes. The weight values can be adjusted to any value as long as the weight value of each node is less than or equal to the weight value of its parent node.

[0027] The categories are arranged in a sequence, such that for any first, second and third categories appearing in the sequence in ascending order (i.e., ... Ci ... Cj ... Ck ..., where Ci is the first category, Cj is the second category, and Ck is the third category), the following inequalities are met: Similarity(Ci, Cj) ≤Similarity(Ci, Ck); and Similarity(Cj, Ck) ≤Similarity(Ci, Ck). A sequence that meets the above inequalities is referred to as a valid sequence. FIG. 3 is a flow diagram illustrating the steps carried out by a sequence generating function to obtain a valid sequence. A valid sequence will be generated for all category nodes by passing the root node to the generation function as the input parameter.

[0028] The sequence generating function begins by assigning the next sequence number assigned by the GetNext function 36 to category X (Step 31). The GetNext function 36 generates sequence numbers sequentially starting from 1. In general, it is allowed to produce any sequential number series which may or may not be contiguous. In Step 32, X is checked for child nodes. If X has child nodes, each child node Y of X recursively calls the sequence generating function (Step 33). If there is more than one child node, the recursive calls are made in series (i.e., one at a time). Otherwise, the program ends (if the sequence generating function was called externally) or the program returns to Step 33 (if the sequence generating function was called by Step 33). In Step 34, a check is made to see if X is a root node. If X is a root node, the program ends. If X is not a root node, the next sequence number is assigned to X (Step 35) and the program returns to Step 33.

[0029] For the category tree shown in FIG. 2, the sequence generating function produces the following sequence: (1, A), (2, B), (3, D), (4, G), (5, D), (6, E), (7, F), (8, B) and (9, C). In Step 32, the sequence generating function visits the child nodes Y in no particular order to recursively call itself. However, if there are other search criteria, it can change that order accordingly. For example, search criteria may be based on popularity of categories. Then, the sequence generating function will visit the child nodes in the order of category popularity, so that child categories with higher user interests are visited before the others. The purpose is to cluster popular categories for efficient searching. As an example, if the popularities of categories D, E and F are in the order F, D and E, the sequence generating function will visit F first, then D and E last. The sequence generated will be (1, A), (2, B), (3, F), (4, D), (5, G), (6, D), (7, E), (8, B) and (9, C).

[0030] Some categories may appear twice in the sequence such as B and D. Each gets assigned two sequence numbers. Any subset of a valid sequence is also a valid sequence. Therefore, the sequence generating function may generate a sequence where each node appears exactly once. As an example, this can be done by skipping either step 31 or 35 in FIG. 3. Then the sequence generated for the example in Fig 1 will be (1, G), (2, D), (3, E), (4, F), (5, B), (6, C) and (7, A) if Steps 31 and 34 are skipped or (1, A), (2, B), (3, D), (4, G), (5, E), (6, F) and (7, C) if Step 35 is skipped. Also, each valid sequence has a reverse sequence where nodes are arranged in reverse order. The corresponding reverse sequence is also a valid sequence.

[0031] Once a valid sequence of nodes is generated in accordance with the sequence generating function of FIG. 3, additional valid sequences can be generated based on the properties of the sequence generation function described above. Indexes of categories are built based on valid sequences and stored in the category database 16. Each index entry corresponds to a category and has two fields: index key and category ID. The index key corresponds to the position of the category in the sequence and the category ID corresponds to the category identifier. For the category tree shown in FIG. 2, the valid sequence generated by the sequence generating function is: (1, A), (2, B), (3, D), (4, G), (5, D), (6, E), (7, F), (8, B) and (9, C), and the index entries are as follows:

A	B	D	G	D	E	F	B	C
1	2	3	4	5	6	7	8	9

where category IDs are in the first row, and corresponding index keys are in the second row.

[0032] Multiple partial indexes may also be stored in the category database 16. Each partial index includes a subset of entries in the full index and includes those categories that meet one or more predefined criteria. The categories that meet the one or more predefined criteria are referred to as qualifying categories. For example, if the criterion is "documents published last week" and only categories B, D, F and C qualify, the partial index has the entries: (2, B), (3, D), (7, F), (8, B) and (9, C). As another example, if the criterion is "the top four categories that have the highest user interests" and categories A, B, D and F meet this criterion, the partial index has the entries: (1, A), (2, B), (3, D), (7, F) and (8, B).

[0033] A mapping table is also stored in the category database 16. The mapping table for the example in FIG. 2 is as follows:

Category ID	Sequence Number
A	1
B	2, 8
C	9
D	3, 5
E	6
F	7
G	4

[0034] A category that is of interest in a particular application is referred to as "an interested category." If the matching of online ads to web page content is desired, the interested category would be the category under which the web page content is classified.

[0035] For an interested category X with a sequence number s, Algorithm A shown in FIG. 4 finds the most relevant category from a set of qualifying categories using the index entries (..., [si, Ci], [sj, Cj], ...) stored in the category database 16. The sequence number s corresponding to the category X is obtained from the mapping table. If X has two sequence numbers assigned to it, then the first sequence number is used. The most relevant category for category X is category Y such that Similarity(X, Y) is the smallest among all Similarity(X, qualifying category).

[0036] Algorithm A begins by searching for X in the set of qualifying categories (Step 41). If X is one of the qualifying categories, then X is identified as the most relevant category. If X is not found in the index, the index is searched in Step 42 for the smallest index range [si, sj], where si < s and s < sj. If s is near the boundary of the index, si and/or sj may not exist. If si does not exist, but sj exists, then Cj is identified as the most relevant category (Step 44a). If sj does not exist, but si exists, then Ci is identified as the most relevant category (Step 44b). If neither si nor sj exists, no category is identified as the most relevant category (Step 44c). If both si and sj exist, flow proceeds to Step 43 where the similarity distance between X and Ci is compared with the similarity distance between X and Cj. If Similarity(Ci, X) < Similarity(Cj, X), Ci is identified as the most relevant category; otherwise, Cj is identified as the most relevant category.

[0037] Steps 41 and 42 may be carried out using conventional range search methods, which is supported by various index implementations either in memory or on disk such as B-tree index. Also, additional search conditions may be placed on the index key range. For example, it may be desired to have the result be within an index key range [a, b].

[0038] FIG. 5 illustrates the nearest category found for each of the interested categories X, Y and Z using Algorithm A. The qualifying categories for building the index are C1, C2, and C3. Node C2 has a very deep descendent node C3. Algorithm A will find C2 as the most relevant category for X even though C2 and C3 have the same similarity distance with X (i.e., the lowest common ancestor of C2 and X is the same as the lowest common ancestor of C3 and X). Likewise, Algorithm A will find C2 as the most relevant category for Y even though C2 and C3 have the same similarity distance with Y (i.e., the lowest common ancestor of C2 and Y is the same as the lowest common ancestor of C3 and Y).

[0039] Steps 42 and 43 of Algorithm A can be applied iteratively to find additional relevant categories. Each time a relevant category is found, that category is excluded from being considered in subsequent iterations.

[0040] FIG. 6 is a flow diagram of Algorithm B that finds N most relevant categories for interested category X with a sequence number s. If X has two sequence numbers assigned to it, the first sequence number is used. Two functions, Prev(s, R) and Next(s, R), are used in Algorithm B, where s is a sequence number; (si, Ci) is an index entry; and R is a set of category IDs (Ci's).

- Prev(s, R) returns the largest index key si such that si $\leq$s and the corresponding Ci is not in R.
- Next(s, R) returns the smallest index key sj such that s $\leq$ sj and the corresponding Cj is not in R.

[0041] In Step 61, R is initialized to an empty set. Then, in Step 62, the index is searched for X. If X is found, it is added to R. In Step 63, the number of categories in R is compared against N. If it is equal to N, this means that N most relevant categories have been found and the program ends. If not, flow proceeds to Step 64, where the functions Prev(s, R) and Next(s, R) are called. For the initial pass through Step 64, s is the sequence number for X. For subsequent passes through Step 64, s is the sequence number assigned in Step 67 or Step 68.

[0042] If both si and sj exist (Step 65), the similarity distance between Ci and X, Similarity(Ci, X), is compared with the similarity distance between Cj and X, Similarity(Cj, X) (Step 66). If Similarity(Ci, X) < Similarity(Cj, X), Ci is added to R and s is assigned the value of si (Step 67). Otherwise, Cj is added to R and s is assigned the value of sj (Step 68). Flow then proceeds to Step 64 after execution of either Step 67 or Step 68.

[0043] The program block in Step 69 is executed if both si and sj do not both exist. If only si exists, flow proceeds to Step 67 where Ci is added to R and s is assigned the value of si. if only sj exists, flow proceeds to Step 68; where Cj is added to R and s is assigned the value of sj. If neither Ci nor Cj exists, the program ends.

[0044] During each pass through the flow diagram of FIG. 6, a relevant category is found, and this category is excluded from being considered in subsequent steps. Algorithm B searches in both directions from the starting number s until it finds N most relevant categories. In finding the N most relevant categories, the similarity distance function is applied at most (N + 1) times.

[0045] The following example illustrates how Algorithm B finds categories C3, C2, C4 and C5 as the most relevant categories and how the index range [si, sj] changes for each pass through the flow diagram of FIG. 6. In the example, the interested category X is C3, and the index entries are (s1, C1), (s2, C2), (s3, C3), (s4, C4), and (s5, C5).

- Pass 1: Find C3
- Pass 2: [si, sj] = [s2, s4] => Find C2
- Pass 3: [si, sj] = [s1, s4] => Find C4
- Pass 4: [si, sj] = [s1, s5] => Find C5

[0046] Algorithm B is very efficient as indicated by the following properties. First, in finding the N most relevant categories, the similarity distance function is applied at most (N + 1) times. Second, it produces category search results in the order of their relevance. The most relevant category is found first. For each new category found, it can also apply other search criteria to qualify the result. It terminates when N categories are found.

[0047] Algorithm B is used to find N most relevant categories to a given category X. If the desired result is to find M documents that are most relevant with respect to category X, Algorithm B is first applied to find the most relevant categories one by one. After each category is found, the category ID of such category is used as a key to retrieve documents within that category. The program terminates when the count of the retrieved documents reaches M.

[0048] FIG. 7 illustrates the process of finding M documents that are most relevant with respect to a given category X in the context of finding M ads that are most relevant to content presented on a web page. In Step 71, the category corresponding to the content presented on a web page is determined. Algorithm B is then executed in Step 72 with the content category as category X. For each relevant category identified by Algorithm B, Steps 73-75 are executed in the order of their relevance until at least M ads are found. Step 73 determines the next relevant category. Step 74 retrieves relevant ads within that category. Step 75 checks to see if at least M relevant ads have been found. If so, the program terminates.

[0049] In the category tree shown in FIG. 2, each node has at most one parent node. For any two nodes, there is a unique lowest common ancestor. There may, however, be situations where a node has more than one parent node. FIG. 8 provides an illustration. As illustrated, F has three parent nodes, A, D and C, while E has two parent nodes, B and C. In situations where nodes of a category tree has more than one parent node, the N most relevant categories that are found using Algorithm B is re-ranked based on overall relevance scores. An overall relevance score between any two nodes, X and Y, takes into account Similarity(X, Y) and other weight factors further described below.

[0050] When a node has more than one parent node, the links between that node and the parent nodes are classified into primary and secondary links. In FIG. 8, primary links are indicated by solid arrows and secondary links are indicated by dotted arrows. Each node has at most one parent node connected via a primary link. All other parent nodes of that node are connected via secondary links. In FIG. 8, D-F is a primary link, and A-F and C-F are secondary links. Also, C-E is a primary link, and B-E is a secondary link.

[0051] The paths from the root to a node are classified into primary and secondary paths. A primary path has only primary links. A secondary path has primary links except for the last link which is a secondary link. Any other path which is neither a primary path nor a secondary path is not considered.

[0052] For the category tree in FIG. 8, the following illustrates how to compute the overall relevance score between F and E. The paths from the root node to F and E are first classified. Paths from A to F include A-B-D-F (Path F1), A-F (Path F2), A-C-F (Path F3), and A-C-D-F (Path F4). Path F1 is a primary path. Paths F2 and F3 are secondary paths. Path F4 is neither a primary path nor a secondary path and is not considered. Paths from A to E include A-C-E (Path E1) and A-B-E (Path E2). Path E1 is a primary path. Path E2 is a secondary path.

[0053] Then, for each path pair, the weight value of the lowest common ancestor of F and E is computed. The value is denoted by score(F, E, pair-i) for pair-i of paths.

- For paths F1 and E1 (pair-1), the lowest common ancestor is A and weight(A) = Similarity(F, E) = 6; then score(F, E, pair-1) = 6.
- For paths F1 and E2 (pair-2), the lowest common ancestor is B and weight(B) = 2; then score(F, E, pair-2) = 2.

- For paths F2 and E1 (pair-3), the lowest common ancestor is A and weight(A) = 6; then score(F, E, pair-3) = 6.
- For paths F2 and E2 (pair-4), the lowest common ancestor is A and weight(A) = 6; then score(F, E, pair-4) = 6.
- For paths F3 and E1 (pair-5), the lowest common ancestor is C and weight(C) = 3; then score(F, E, pair-5) = 3.
- For paths F3 and E2 (pair-6), the lowest common ancestor is A and weight(A) = 6; then score(F, E, pair-6) = 6.

[0054] The overall relevance factor is a scoring function combining score(F, E, pair-i) for all i. For example, the following scoring function may be used:

Overall-Relevance-Score(F, E) = $\sum f(i) * (S - \text{score}(F, E, \text{pair-}i)) = f(1) * (S - 6) + f(2) * (S - 2) + f(3) * (S - 6) + f(4) * (S - 6) + f(5) * (S - 3) + f(6) * (S - 6)$;

where, f(i) and S are configurable constants. For example, if S = 7, f(1) = 2, and f(i) = 1 for i > 1, then the overall relevance score = 14. A larger overall relevance score indicates a higher degree of relevance between the two categories.

[0055] In the category tree shown in FIG. 2, the weight value of each node is less than or equal to the weight value of its parent node. In such situations, the category that is most relevant to an interested category X is determined using Algorithm A. There may, however, be situations where the weight values of some nodes are greater than the weight value of their respective parent node. These nodes are referred to as "pivot" nodes. In these situations, the pivot nodes that are ancestor nodes of X are identified and the category tree is divided into different regions. FIG. 9 provides an illustration. The first region #A includes the lowest pivot ancestor node and all of its descendent nodes. The second region #B includes the next lowest pivot ancestor node and all of its descendent nodes except the nodes in region #A. The third region #C includes all nodes that are not in the first region #A or the second region #C. In this example, the category tree is divided into three regions because there are two pivot nodes that are ancestor nodes to X. In general, if there are N pivot nodes that are ancestor nodes to X, the category tree will be divided into N+1 regions.

[0056] Following the division, the category that is most relevant to X is found using the following logic:

- Find the nearest category to X in area #A. If this is C1, then Similarity(X, C1) $\leq$ Similarity(X, Y) for any node Y in area #A.
- Find the nearest category to X in area #B. If this is C2, where Similarity(X, C2) $\leq$ Similarity(X, Y) for any node Y in area #B.
- Find the nearest category to X in area #C. If this is C3, where Similarity(X, C3) $\leq$ Similarity(X, Y) for any node Y in area #C.
- Compare Similarity(X, C1), Similarity(X, C2), and Similarity(X, C3). The category with the smallest similarity distance with X will be category that is most relevant to X.

[0057] Areas #A, #B, and #C correspond to a key range in the index used for search. Key ranges are determined by the sequence numbers of P1 and P2 derived from sequence generating function of FIG. 3. If P1 gets assigned sequence numbers a and b, where a < b, and P2 gets assigned sequence numbers c and d, where c < d, then:

- Area #A has the key range = { key | key $\geq$ a and key $\leq$ b}.
- Area #B has the key range = { key | key $\geq$ c and key $\leq$ d and key < a and key > b}.
- Area #C has the key range = { key | key < c and key > d}.

[0058] Once the three key ranges are identified, Algorithm A is used to find the nearest category within each index key range identified above. The nearest category may not exist in a key range if the index used for search has no key values within that range. FIG. 10 is a flow diagram illustrating the process steps carried out by Algorithm E to find most relevant category for category X. The process steps of FIG. 10 are described below:

- Step 91. Determine all pivot ancestor nodes of category X. They are: Pi, i = 1, 2, ..., k, where Pj is an ancestor of Pi if i < j.
- Step 92. For each pivot ancestor node, Pi, i = 1, 2, ··· k, find its two sequence numbers, ai and bi (ai ≤ bi), using the mapping table.
- Step 93. Find the nearest category to X within the key range = { key | key ≥ a1 and key ≤ b1}. The nearest category is the category within this key range that has the smallest similarity distance to X.
- Step 94. Find the nearest category to X within the key range = { key | key ≥ ai+1 and key ≤ bi+1 and key < ai and key > bi} for i = 1, 2, ..., k-1. The nearest category is the category within the indicated key range that has the smallest similarity distance to X.
- Step 95. Find the nearest category to X within the key range = { key | key < ak and key > bk}. The nearest category is the category within this key range that has the smallest similarity distance to X.
- Step 96. For all nearest categories Yj found in steps 93, 94 and 95, compare Similarity(X, Yj). The category Y out of all Yj with the smallest similarity distance to X is the result, i.e., Similarity(X, Y) ≤ Similarity(X, Yj) for all j.

[0059] The key range in Step 94 consists of two disjoint sub-ranges { key | key $\geq$ ai+1 and key < ai } and { key | key > bi and key $\leq$ bi+1 }. Within each sub-range, the first sequence number of X is used as the corresponding search key value to search for the nearest category. Similarly, in Step 95 we use the first sequence number of X to search for the nearest category within each sub-range. Also, Step 96 can be replaced by using a priority queue in Steps 93, 94, and 95 to track Yj with the smallest similarity distance to X.

[0060] In the embodiments of the invention described above, each node of the category tree may be a single category or a cluster of categories. Each cluster has a valid category sequence and an index defined for the categories in its cluster. When there is a large number of category nodes, clustering of categories and nested search techniques allows for efficient search of relevant categories. For example, if N most relevant categories are desired and the most relevant category is a cluster that has more than N categories, Algorithm B can be executed using the category sequence and index defined for the categories of the cluster to find the N categories that are most relevant.

[0061] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.

What is claimed is:

1. In a document management system in which documents are classified into categories, a method of searching for documents that are most relevant to certain content, the method comprising the steps of:
generating a category sequence in which a category appears one or more times in the category sequence;
determining a category for said content and a position of said content category in the category sequence;
identifying a first category that is directly prior to the position of the content category in the category sequence;
identifying a second category that is directly after the position of the content category in the category sequence;
determining a first similarity distance between the first category and the content category and a second similarity distance between the second category and the content category; and
identifying, as the documents most relevant to said content, (i) the documents that are classified in the content category if the content category is part of the category sequence; (ii) the documents that are classified in the first category if the first similarity distance is less than the second similarity distance; and (iii) the documents that are classified in the second category if the first similarity distance is greater than or equal to the second similarity distance.

2. The method according to claim 1, further comprising the steps of:
identifying additional categories that are nearest the position of the content category in the category sequence; and
identifying those documents that are classified in the additional categories also as documents that are most relevant to said content.

3. The method according to claim 1, wherein, for any first, second and third categories appearing in the category sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories.

4. The method according to claim 1, wherein the categories are related to one another in a hierarchical manner and each category is assigned a weight value, and wherein the similarity distance between two categories is based on the weight value of the lowest common ancestor, such that a larger weight value results in a larger similarity distance.

5. The method according to claim 4, wherein the weight value for a category is assigned based on the number of categories that descend from said category.

6. The method according to claim 4, wherein the weight value for a category is always greater than or equal to the weight value of any category that descends from said category.

7. The method according to claim 4, wherein at least one of the categories has a weight value that is greater than a weight value of its parent node.

8. The method according to claim 1, wherein the category sequence includes all of the categories.

9. The method according to claim 1, wherein the category sequence includes only some of the categories.

10. The method according to claim 1, wherein the documents comprise online ads and the content comprise web page content.

11. A method of indexing categories for subsequent retrieval, comprising the steps of:

generating a sequence of categories, wherein, for any first, second and third categories appearing in the sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories; and

storing for each category an identifier for said category and an associated index key identifying a position of said category in said sequence.

12. The method according to claim 11, wherein the categories are related to one another in a hierarchical manner and each category is assigned a weight value, and wherein the similarity distance between two categories is based on the weight value of the lowest common ancestor.

13. The method according to claim 12, wherein the weight value for a category is assigned based on the number of categories that descend from said category.

14. The method according to claim 12, wherein the weight value for a category is always greater than or equal to the weight value of any category that descends from said category.

15. The method according to claim 12, wherein at least one of the categories has a weight value that is greater than a weight value of its parent node.

16. The method according to claim 11, wherein at least one of the categories is a category cluster including multiple categories.

17. The method according to claim 16, further comprising the steps of:

generating a sequence from the multiple categories, wherein, for any first, second and third categories appearing in the sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories; and

storing for each category an identifier for said category and an associated index key identifying a position of said category in said sequence generated from the multiple categories.

18. A document management system in which documents are classified into a plurality of categories, comprising:

a category database having a plurality of category records, each category record including an identifier for said category and a position of said category in a category sequence, wherein, for any first, second and third categories appearing in the category sequence in ascending order, a similarity distance between the first and second categories is less than or equal to a similarity distance between the first and third categories, and a similarity distance between the second and third categories is also less than or equal to the similarity distance between the first and third categories; and

a processor programmed to generate a set of categories most relevant to a reference category using the category database.

19. The document management system according to claim 18, wherein the categories are related to one another in a hierarchical manner and each category is assigned a weight value, and wherein the similarity distance between two categories is based on the weight value of the lowest common ancestor.

20. The document management system according to claim 19, wherein the weight value for a category is assigned based on the number of categories that descend from said category.

21. The document management system according to claim 19, wherein the weight value for a category is always greater than or equal to the weight value of any category that descends from said category.

22. The document management system according to claim 19, wherein at least one of the categories has a weight value that is greater than a weight value of its parent node.

23. The document management system according to claim 18, wherein the category database comprises a full index representative of all categories of the document management system.

24. The document management system according to claim 23, wherein the category database further comprises a partial index representative of some of the categories of the document management system, and wherein the processor is further programmed to generate multiple partial indexes from the full index based on predefined criteria.

25. In a category indexing system, wherein categories are related to one another in a hierarchical manner and each category is assigned a weight value, and wherein the similarity distance between two categories is based on the weight value of the lowest common ancestor, a method of identifying a category that is more relevant to a reference category, comprising the steps of:

identifying at least one ancestor node of the reference category that has a weight value that is greater than a weight value of a parent node of said ancestor node;

determining a first category in a first set of categories that is nearest to said reference category;

determining a second category in a second set of categories that is nearest to said reference category; and comparing a similarity distance between the first category and the reference category and a similarity distance between the second category and the reference category, wherein the one with the smaller similarity distance is the more relevant category.

26. The method according to claim 25, further comprising the steps of generating a sequence of categories and defining the first and second sets of categories.

27. The method according to claim 26, wherein the first set includes all of the categories that descend from the ancestor node and the second set does not include any categories in the first set.

FIG. 1

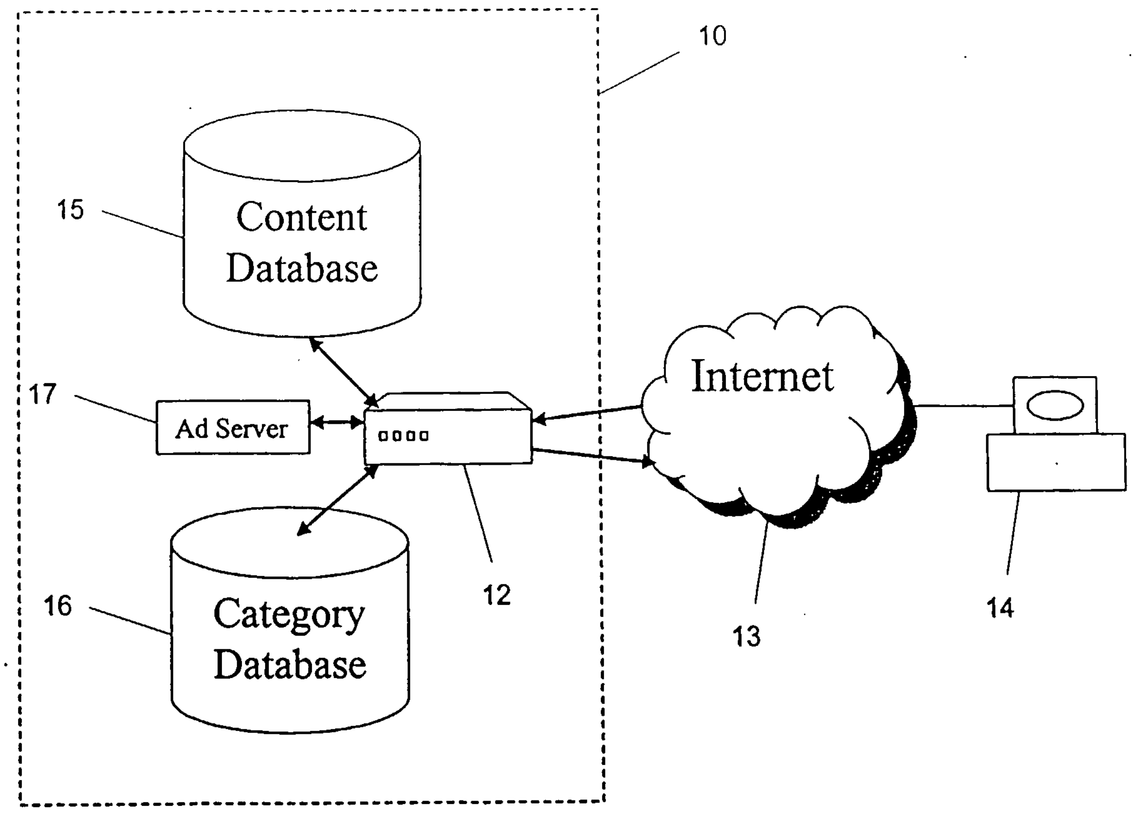

10
15
Content Database
17
Ad Server
12
16
Category Database
Internet
13
14

FIG. 2

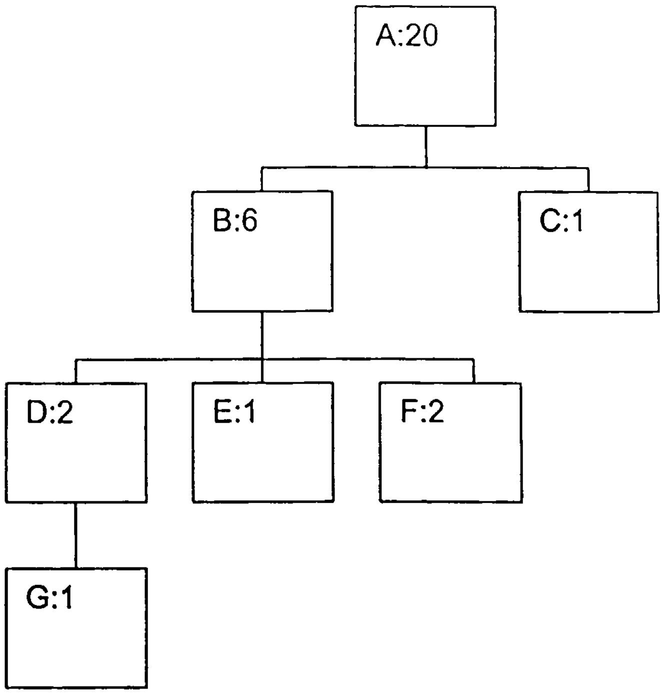


FIG. 3

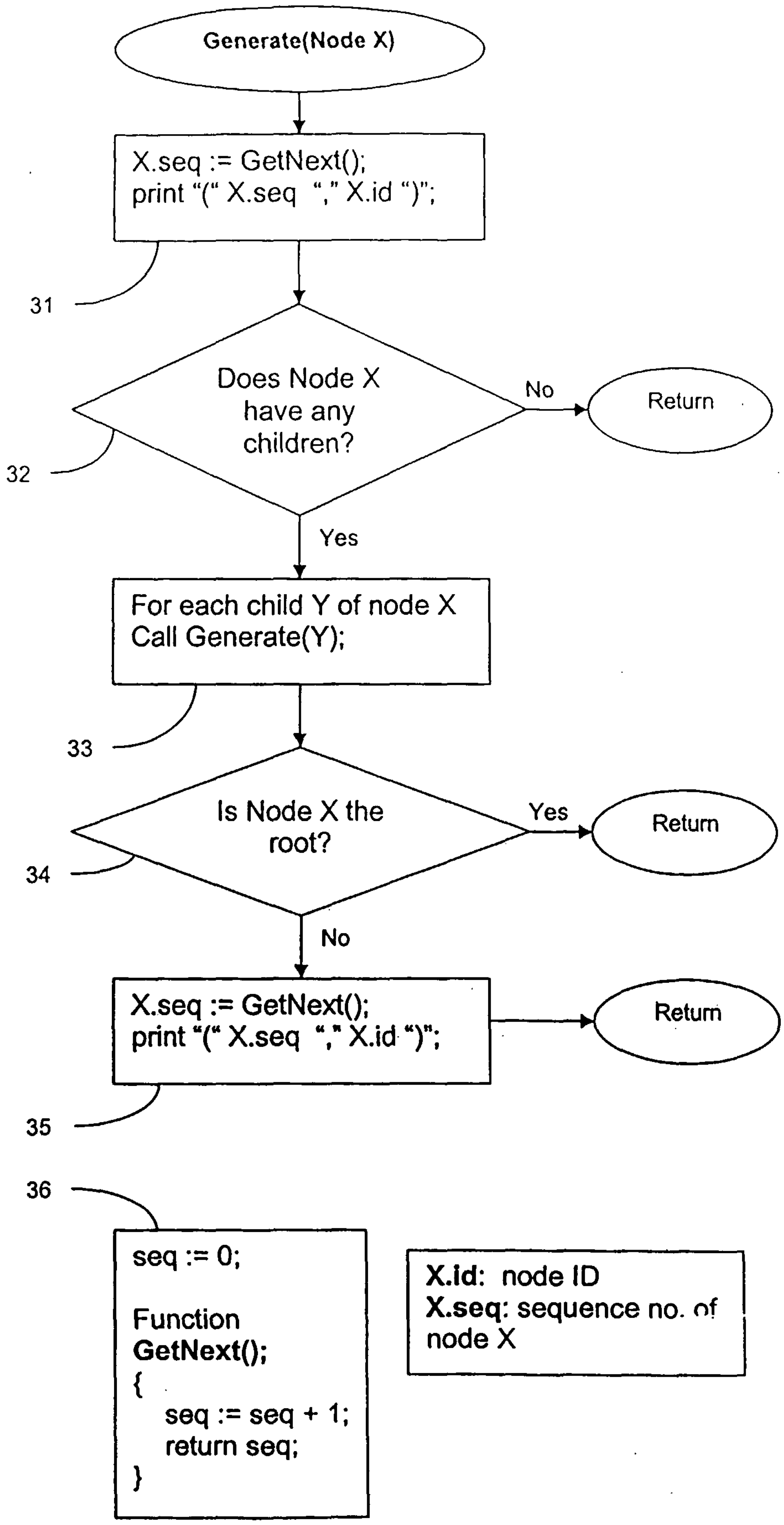

Generate(Node X)
X.seq := GetNext();
print "(" X.seq "," X.id ")";
31
Does Node X have any children?
No
Return
32
Yes
For each child Y of node X
Call Generate(Y);
33
Is Node X the root?
Yes
Return
34
No
X.seq := GetNext();
print "(" X.seq "," X.id ")";
Return
35
36
seq := 0;
Function GetNext();
{
seq := seq + 1;
return seq;
}
X.id: node ID
X.seq: sequence no. of node X

FIG. 4

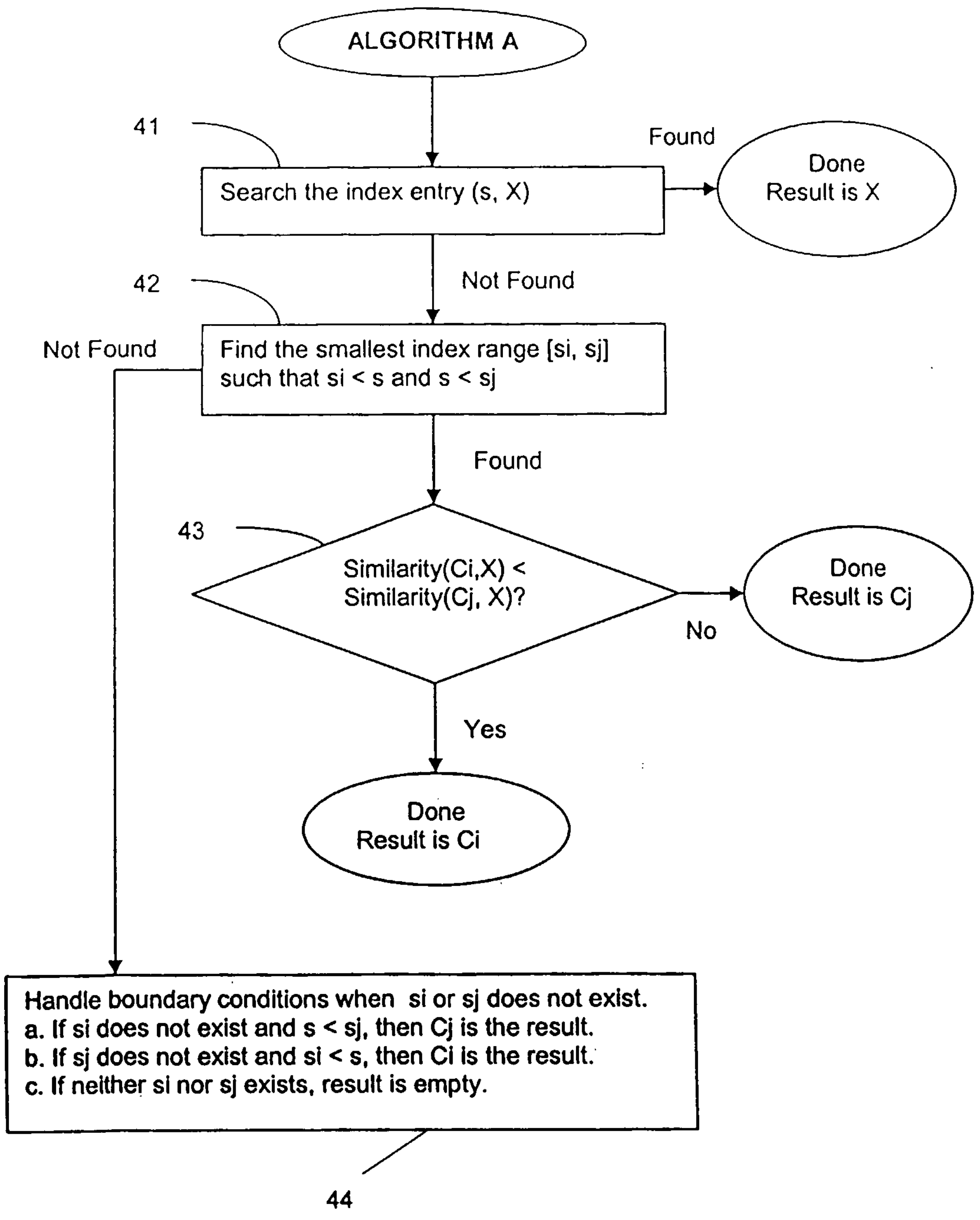
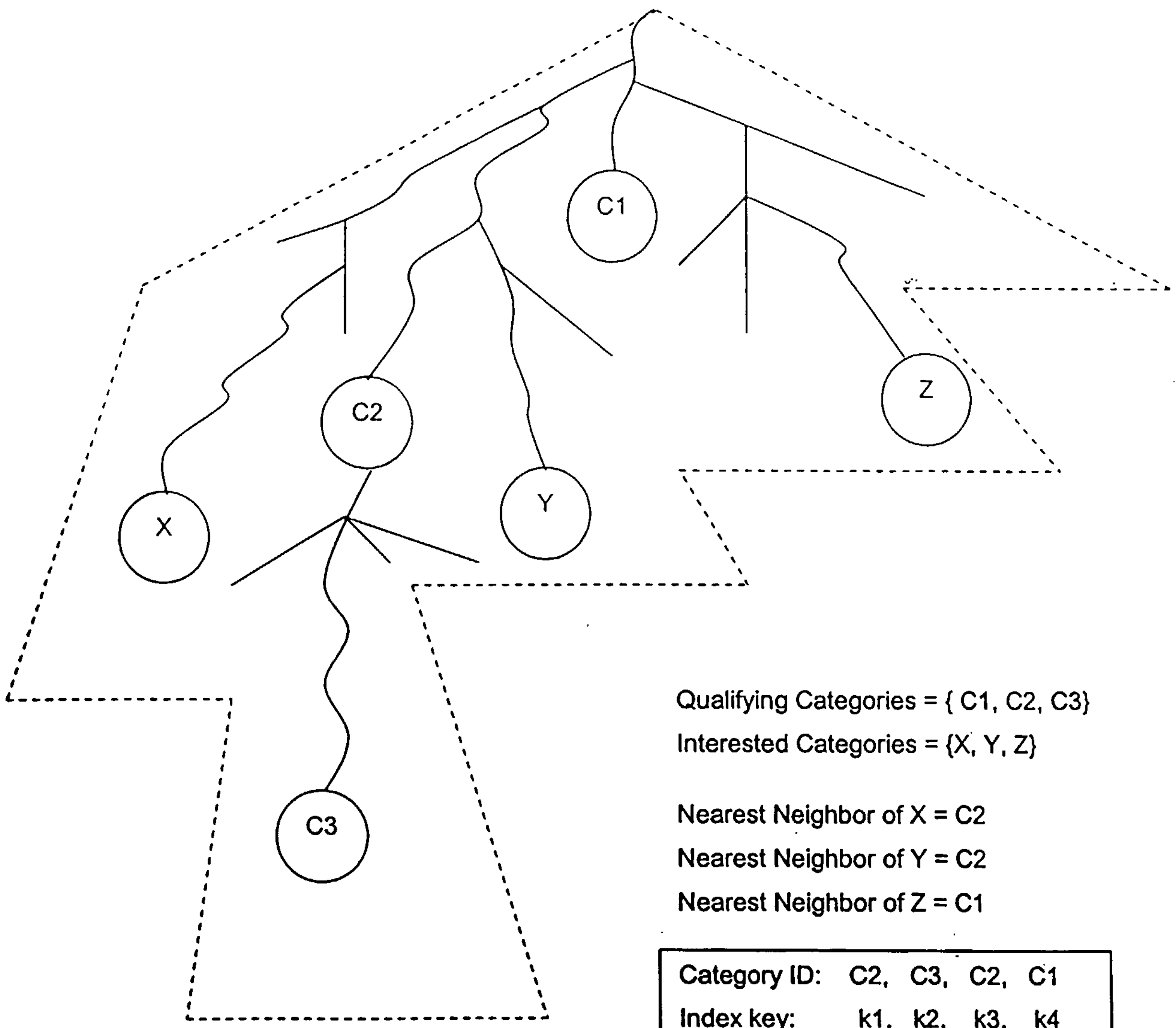

ALGORITHM A
41
Search the index entry (s, X)
Found
Done
Result is X
Not Found
42
Not Found
Find the smallest index range [si, sj]
such that si < s and s < sj
Found
43
Similarity(Ci,X) <
Similarity(Cj, X)?
No
Done
Result is Cj
Yes
Done
Result is Ci
Handle boundary conditions when si or sj does not exist.
a. If si does not exist and s < sj, then Cj is the result.
b. If sj does not exist and si < s, then Ci is the result.
c. If neither si nor sj exists, result is empty.
44

FIG. 5



Qualifying Categories = { C1, C2, C3}

Interested Categories = {X, Y, Z}

Nearest Neighbor of X = C2

Nearest Neighbor of Y = C2

Nearest Neighbor of Z = C1

Category ID:	C2,	C3,	C2,	C1
Index key:	k1,	k2,	k3,	k4

FIG. 6

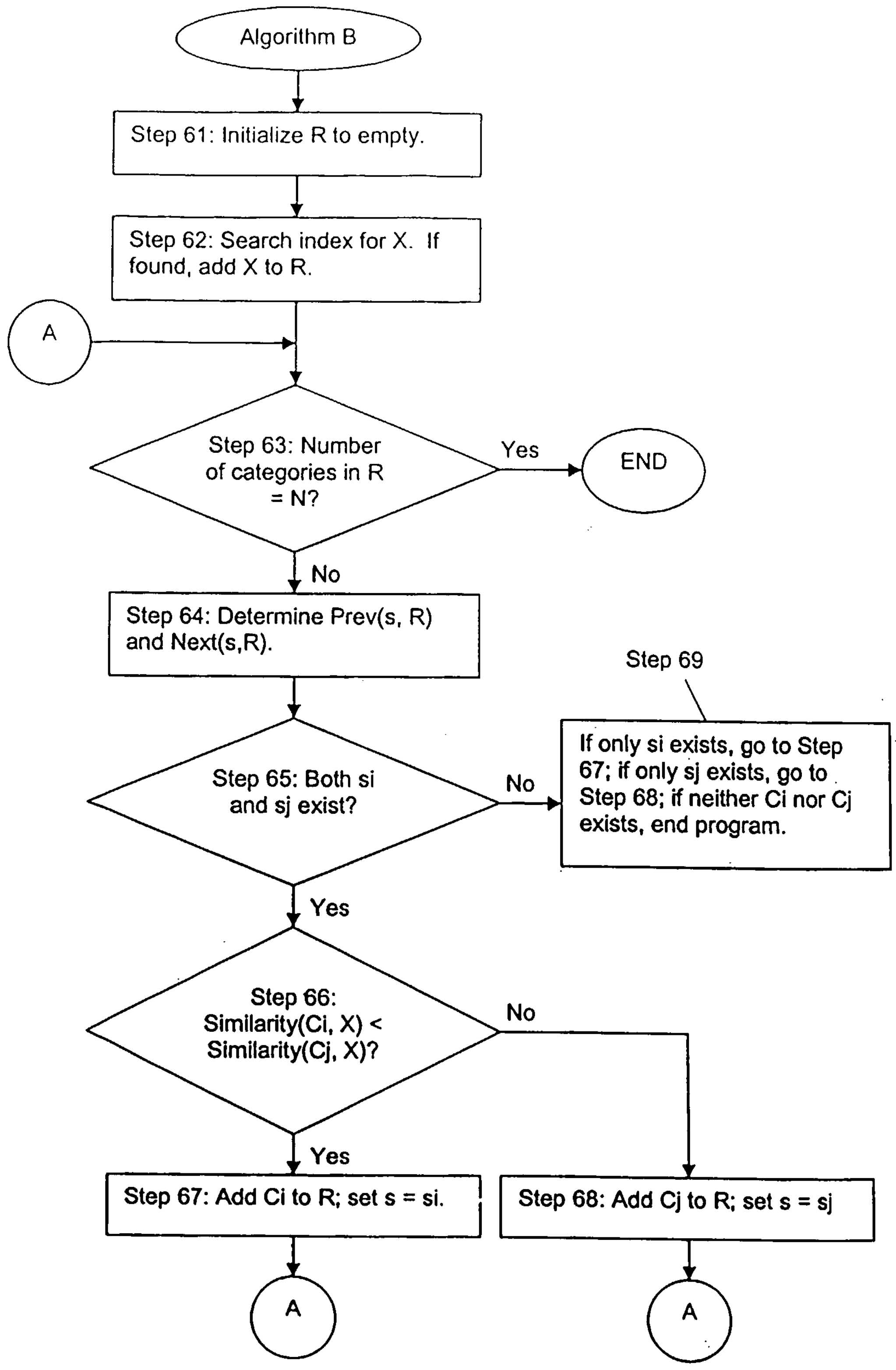

Algorithm B
Step 61: Initialize R to empty.
Step 62: Search index for X. If found, add X to R.
A
Step 63: Number of categories in R = N?
Yes
END
No
Step 64: Determine Prev(s, R) and Next(s,R).
Step 69
If only si exists, go to Step 67; if only sj exists, go to Step 68; if neither Ci nor Cj exists, end program.
Step 65: Both si and sj exist?
No
Yes
Step 66: Similarity(Ci, X) < Similarity(Cj, X)?
No
Yes
Step 67: Add Ci to R; set s = si.
Step 68: Add Cj to R; set s = sj
A
A

FIG. 7

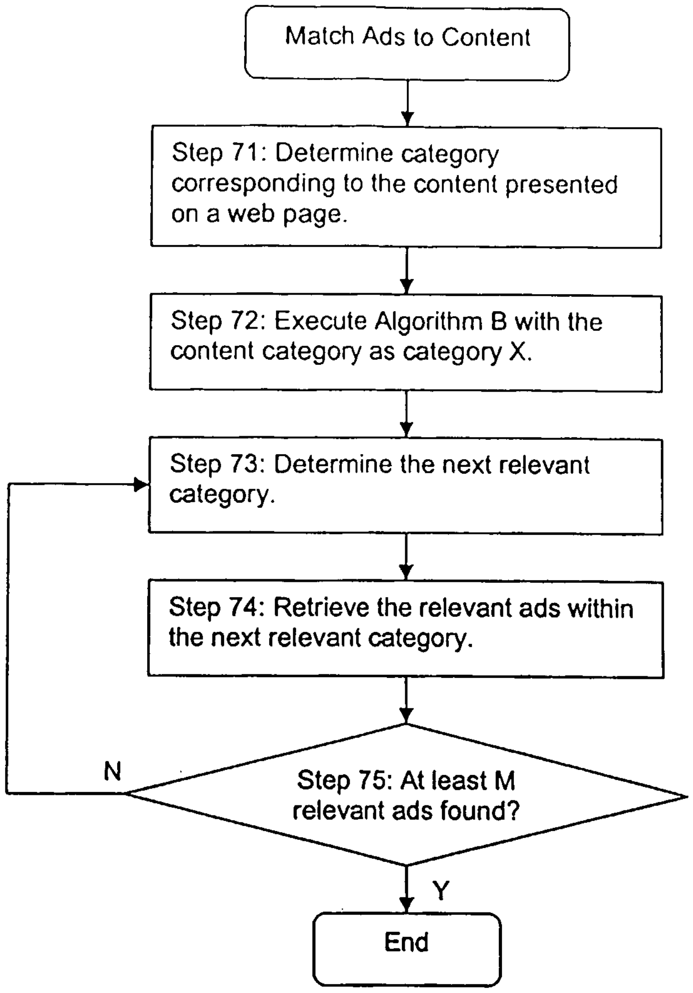

Match Ads to Content
Step 71: Determine category corresponding to the content presented on a web page.
Step 72: Execute Algorithm B with the content category as category X.
Step 73: Determine the next relevant category.
Step 74: Retrieve the relevant ads within the next relevant category.
Step 75: At least M relevant ads found?
N
Y
End

FIG. 8

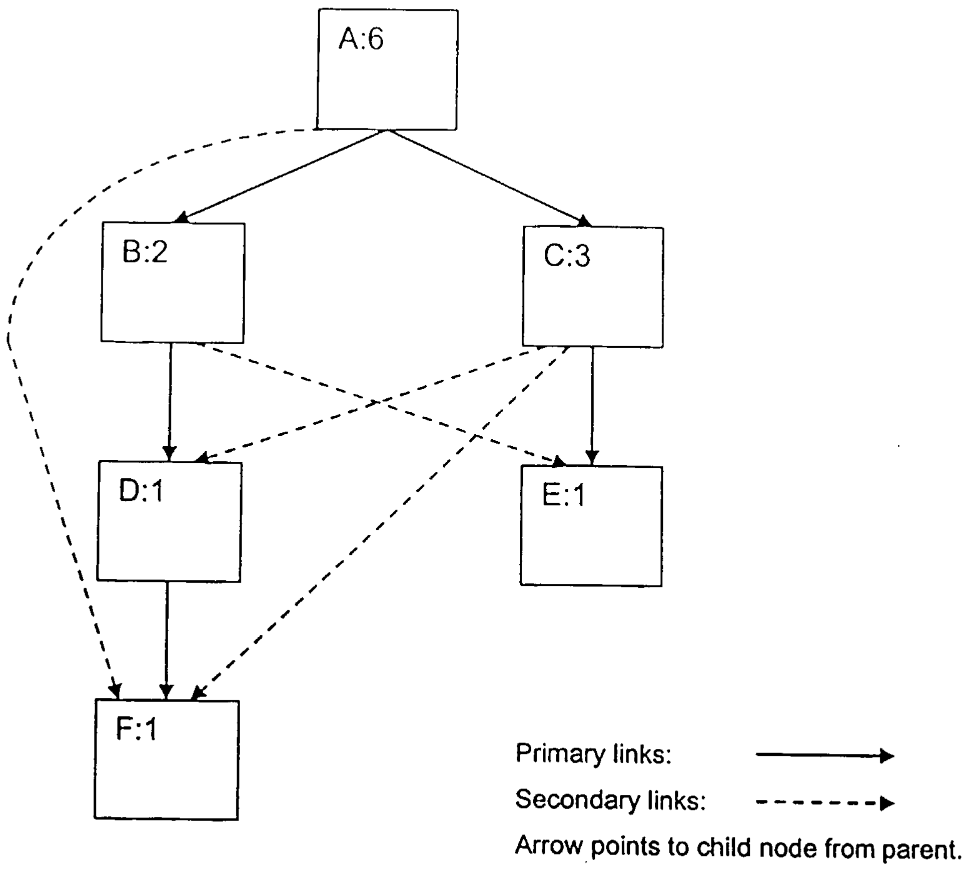
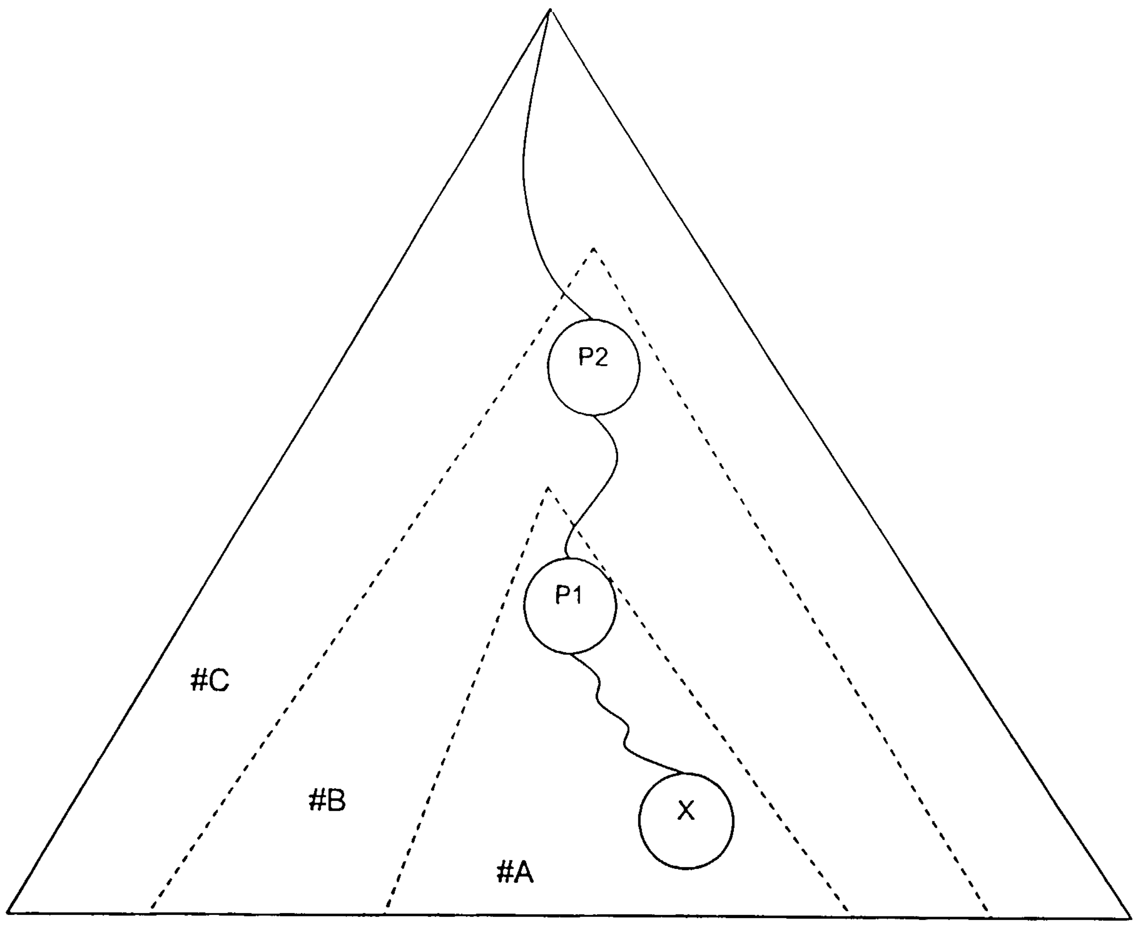
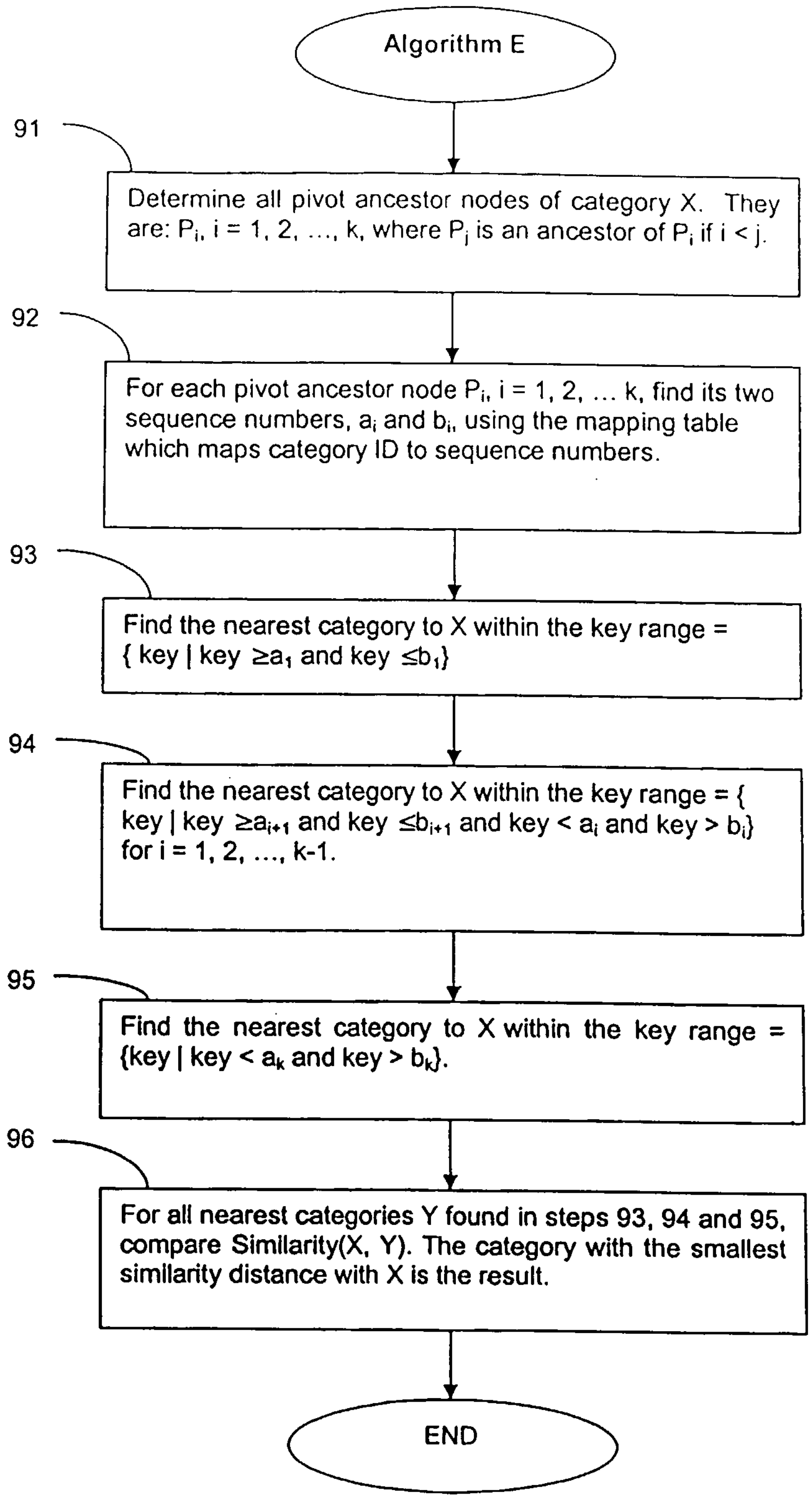

A:6
B:2
C:3
D:1
E:1
F:1
Primary links:
Secondary links:
Arrow points to child node from parent.

FIG. 9



Only interested node X and its pivot ancestors P1 and P2 are shown here. There may be other branches and nodes in the tree and along the path from the root to node X.
Pivot nodes P1 and P2 divide the tree into three areas #A, #B, and #C.

FIG. 10



Algorithm E
91
Determine all pivot ancestor nodes of category X. They are: P_i, i = 1, 2, ..., k, where P_j is an ancestor of P_i if i < j.
92
For each pivot ancestor node P_i, i = 1, 2, ... k, find its two sequence numbers, a_i and b_i, using the mapping table which maps category ID to sequence numbers.
93
Find the nearest category to X within the key range = { key | key $\geq a_1$ and key $\leq b_1$}
94
Find the nearest category to X within the key range = { key | key $\geq a_{i+1}$ and key $\leq b_{i+1}$ and key < a_i and key > b_i} for i = 1, 2, ..., k-1.
95
Find the nearest category to X within the key range = {key | key < a_k and key > b_k}.
96
For all nearest categories Y found in steps 93, 94 and 95, compare Similarity(X, Y). The category with the smallest similarity distance with X is the result.
END