PROPAGATING RELEVANCE FROM LABELED DOCUMENTS TO UNLABELED DOCUMENTS

BACKGROUND

[0001] Many search engine services, such as Google and Overture, provide for searching for information that is accessible via the Internet. These search engine services allow users to search for display pages, such as web pages, that may be of interest to users. After a user submits a search request (i.e., a query) that includes search terms, the search engine service identifies web pages that may be related to those search terms. To quickly identify related web pages, the search engine services may maintain a mapping of keywords to web pages. This mapping may be generated by "crawling" the web (i.e., the World Wide Web) to identify the keywords of each web page. To crawl the web, a search engine service may use a list of root web pages to identify all web pages that are accessible through those root web pages. The keywords of any particular web page can be identified using various well-known information retrieval techniques, such as identifying the words of a headline, the words supplied in the metadata of the web page, the words that are highlighted, and so on. The search engine service identifies web pages that may be related to the search request based on how well the keywords of a web page match the words of the query. The search engine service then displays to the user links to the identified web pages in an order that is based on a ranking that may be determined by their relevance to the query, popularity, importance, and/or some other measure.

[0002] Three well-known techniques for ranking of web pages are PageRank, HITS ("Hyperlinked-Induced Topic Search"), and DirectHIT. PageRank is based on the principle that web pages will have links to (i.e., "outgoing links") important web pages. Thus, the importance of a web page is based on the number and importance of other web pages that link to that web page (i.e., "incoming links"). In a simple form, the links between web pages can be represented by adjacency matrix A, where A_{ij} represents

the number of outgoing links from web page i to web page j. The importance score w_j for web page j can be represented by the following equation:

$$w_j = \Sigma_i A_{ij} w_i$$

[0003] This equation can be solved by iterative calculations based on the following equation:

$$A^T w = w$$

where w is the vector of importance scores for the web pages and is the principal eigenvector of A^T.

[0004] The HITS technique is additionally based on the principle that a web page that has many links to other important web pages may itself be important. Thus, HITS divides "importance" of web pages into two related attributes: "hub" and "authority." "Hub" is measured by the "authority" score of the web pages that a web page links to, and "authority" is measured by the "hub" score of the web pages that link to the web page. In contrast to PageRank, which calculates the importance of web pages independently from the query, HITS calculates importance based on the web pages of the result and web pages that are related to the web pages of the result by following incoming and outgoing links. HITS submits a query to a search engine service and uses the web pages of the result as the initial set of web pages. HITS adds to the set those web pages that are the destinations of incoming links and those web pages that are the sources of outgoing links of the web pages of the result. HITS then calculates the authority and hub score of each web page using an iterative algorithm. The authority and hub scores can be represented by the following equations:

$$a(p) = \sum_{q \to p} h(q) \text{ and } h(p) = \sum_{p \to q} a(q)$$

where $a(p)$ represents the authority score for web page p and $h(p)$ represents the hub score for web page p. HITS uses an adjacency matrix A to represent the links. The adjacency matrix is represented by the following equation:

$$b_{ij} = \begin{cases} 1 \text{ if page } i \text{ has a link to page } j, \\ 0 \text{ otherwise} \end{cases}$$

[0005] The vectors a and h correspond to the authority and hub scores, respectively, of all web pages in the set and can be represented by the following equations:

$$a = A^T h \text{ and } h = Aa$$

Thus, a and h are eigenvectors of matrices $A^T A$ and AA^T. HITS may also be modified to factor in the popularity of a web page as measured by the number of visits. Based on an analysis of click-through data, b_{ij} of the adjacency matrix can be increased whenever a user travels from web page i to web page j.

[0006] DirectHIT ranks web pages based on past user history with results of similar queries. For example, if users who submit similar queries typically first selected the third web page of the result, then this user history would be an indication that the third web page should be ranked higher. As another example, if users who submit similar queries typically spend the most time viewing the fourth web page of the result, then this user history would be an indication that the fourth web page should be ranked higher. DirectHIT derives the user histories from analysis of click-through data.

[0007] Some ranking techniques use machine learning algorithms to learn a ranking function from training data that includes queries, feature vectors representing pages, and for each query, a ranking for each page. A ranking function serves as a mapping from features of a page to its rank for a given query. The learning of a ranking function has been considered by some as a regression problem for learning the mapping of a feature vector to a member of an ordered set of numerical ranks. Some regression based techniques attempt to provide an absolute relevance score that can be used to rank pages. A ranking function, however, need not provide an absolute relevance score but rather need only provide a relative ranking of the pages. Thus, these regression-based techniques solve a problem that is more difficult than needed.

[0008] Machine learning algorithms for a ranking function use queries, feature vectors, and user-labeled relevance scores as training data. To generate the training

data, queries may be submitted to a search engine which generates the pages of the search result. The algorithms then generate the feature vectors for the pages and input from a user the relevance scores for each page. A difficulty with such an approach is that a search engine may return hundreds of pages as its search result. It can be quite costly to have a user label all the pages of a search result. Moreover, it can be difficult for a user to accurately assess the relevance of such a large number of pages. Although a user could label only a small portion of the pages, the learning based on such a small portion may not provide an accurate ranking function.

SUMMARY

[0009] A method and system for propagating the relevance of labeled documents to a query to the relevance of unlabeled documents is provided. The propagation system provides training data that includes queries, documents labeled with their relevance to the queries, and unlabeled documents. The propagation system then calculates the similarity between pairs of documents in the training data. The propagation system then propagates the relevance of the labeled documents to similar, but unlabeled, documents. The propagation system may iteratively propagate labels of the documents until the labels converge on a solution. The training data with the propagated relevances can then be used to train a ranking function.

[0010] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Figure 1 is a diagram that illustrates a portion of a graph of documents.

[0012] Figure 2 is a block diagram that illustrates components of the propagation system in one embodiment.

[0013] Figure 3 is a flow diagram that illustrates the processing of the create ranking function component of the propagation system in one embodiment.

[0014] Figure 4 is a flow diagram that illustrates the processing of the propagate relevance component of the propagation system in one embodiment.

[0015] Figure 5 is a flow diagram that illustrates the processing of the build graph component of the propagation system in one embodiment.

[0016] Figure 6 is a flow diagram that illustrates the processing of the generate weights for graph component of the propagation system in one embodiment.

[0017] Figure 7 is a flow diagram that illustrates the processing of the normalize weights of graph component of the propagation system in one embodiment.

[0018] Figure 8 is a flow diagram that illustrates the processing of the propagate relevance based on graph component of the propagation system in one embodiment.

DETAILED DESCRIPTION

[0019] A method and system for propagating relevance of labeled documents to a query to unlabeled documents is provided. In one embodiment, the propagation system provides training data that includes queries, documents (represented by feature vectors) labeled with their relevance to the queries, and unlabeled documents. For example, the propagation system may submit a query to a search engine and use the search result as the documents (e.g., web pages). The propagation system may then prompt a user to label some of the documents of the search result based on their relevance to the query. The propagation system then calculates the similarity between pairs of documents in the training data. For example, the propagation system may represent each document by a feature vector and may calculate the similarity between documents based on the Euclidean distance in feature space or based on a cosine similarity metric. The propagation system then propagates the relevance of the labeled documents to similar, but unlabeled, documents. The propagation system may iteratively propagate labels of the documents until the labels converge on a solution. The training data with the propagated relevances can then be used to train a ranking function. In this way, the propagation system can automatically augment training data with additional training data based on similarities between documents.

[0020] In one embodiment, the propagation system represents the documents using a document graph with each node representing a document and each edge representing similarity between the documents represented by the connected nodes. The propagation system may represent the graph as a square matrix with a row and column for each document in which each non-zero value indicates an edge between the node of the row and the node of the column. The propagation system may define edges for the graph using various techniques. For example, the propagation system may consider the graph to be fully connected in which case each node has an edge to every other node. As another example, the propagation system may consider the nodes to be connected via a minimum spanning tree. In one embodiment, the propagation system considers the nodes to be connected using a k-nearest neighbor algorithm. In particular, the propagation system identifies the k nearest neighbors to each node and adds an edge from that node to each of its k nearest neighbors. The propagation system then calculates weights for the edges based on the similarity between the documents represented by the connected edges. The propagation system may use various techniques for determining the similarity between documents. In one embodiment, the propagation system uses a Euclidean distance metric based on the feature vector representation of the documents in a feature space. The propagation system stores the similarity as the values of the square matrix resulting in a similarity or affinity matrix. The propagation system may also normalize the similarity matrix. The propagation system may also set the diagonal values to 0 to prevent self-reinforcement during relevance propagation.

[0021] After generating the similarity matrix, the propagation system propagates the relevance of the labeled documents to the unlabeled documents using a manifold ranking based propagation algorithm. A manifold ranking based algorithm is described in He, J., Li, M., Zhang, H.J., et al., "Manifold-Ranking Based Image Retrieval," Proc. of the 12th Annual ACM International Conf. on Multimedia, 2004. The propagation system initially sets the relevance of the labeled documents to the relevance score provided by the user and the relevance of the unlabeled documents to 0. The propagation system then spreads the relevance of the labeled documents to their connected unlabeled documents factoring in the similarity as indicated by the similarity matrix. The propagation system iteratively spreads the relevance score until the relevance scores

converge on a solution. The resulting relevance scores of the unlabeled documents will be in proportion to the probability that they are relevant to the same query as the labeled documents. An unlabeled document that is very similar to many labeled documents with high relevance scores will thus have a high relevance score. Conversely, an unlabeled document that is not very similar to any labeled documents will have a low relevance score.

[0022] The propagation system may represent similarity using a Laplace kernel, which can be represented by the following equation:

$$k_L(x_i, x_j) = \prod_{l=1}^{t} \frac{1}{2\sigma_l} \exp\left(-\left|x_{il} - x_{jl}\right| / \sigma_l\right) \quad (1)$$

where x_{il} and x_{jl} represent the l^{th} dimension of x_i and x_j respectively, t represents the dimensionality of the feature space, and σ_l represents a positive parameter that reflects the weights of different dimensions in the similarity calculation. Thus, the propagation system represents the weight of the edges by the following equation:

$$W_{ij} = k_L(x_i, x_j) = \prod_{l=1}^{t} \exp\left(-\left|x_{il} - x_{jl}\right| / \sigma_l\right) \quad (2)$$

where W_{ij} represents the similarity between documents i and j. The propagation system may omit the constant coefficient $1/2\sigma_l$ since its effect on the similarity matrix W will be counteracted by the normalization of the matrix. The propagation system normalizes the similarity matrix as represented by the following equation:

$$S = D^{-1/2} W D^{-1/2} \quad (3)$$

where S represents the normalized similarity matrix and D represents a diagonal matrix where (i,i) is equal to the sum of the ith row of similarity matrix W. The normalization normalizes the similarities to be relative to the similarity of the connected documents.

[0023] The propagation system may represent each document as a t dimension feature vector x that forms a point in the Euclidian space. For a query, the

propagation system receives the result set of documents $\chi = \{x_{l1}, x_{l2}, \ldots x_{lm}, x_{u1}, x_{u2}, \ldots x_{un}\} \subset \square^{'}$. The first m points (in feature space) represent user-labeled documents, and the last n points (in feature space) represent unlabeled documents. The propagation system also receives a corresponding label vector $\gamma = [y_{l1}, y_{l2}, \ldots, y_{lm}, 0, 0, \ldots, 0]^T$. The last n labels have the value of 0 to represent unlabeled documents. The propagation system may also allow for the specification of negative labels, rather than only positive labels, to represent negative examples of relevance. The propagation system represents distance between documents in feature space as $d : \chi \times \chi \to \square$, which assigns each pair of points x_i and x_j a distance $d(x_i, x_j)$, and represents a ranking function of the documents as $f : \chi \to \square$, which assigns to each point x_i a ranking score f_i. The ranking function learning problem is to learn $f : \chi \to \square$ from a set of queries with the features $X = \{\chi_q\}$ and the labels $Y = \{\gamma_q\}$. The propagation system represents the limit of the relevance propagation by the following equation:

$$f^* = (1-\alpha)(I-\alpha S)^{-1} y \tag{4}$$

where f^* represents the limit of the relevance, y represents the initial labels, and α represents a decay factor. Because it is computationally difficult to calculate the inverse of the normalized similarity matrix S, the propagation system approximates f^* using a Taylor series expansion. The propagation system may represent the Taylor series expansion by the following equation:

$$\begin{aligned} f^* &= (I-\alpha S)^{-1} y \\ &= (I+\alpha S+\alpha^2 S^2+\ldots) y \\ &= y+\alpha S y+\alpha S(\alpha S y)+\ldots \end{aligned} \tag{5}$$

The propagation system iteratively solves for f^* until it converges on a solution or for a fixed number of iterations.

[0024] Once the relevances are propagated, the propagation labeled system may use the training data sets (query and labeled feature vectors) to train a ranking function. A ranking function may be implemented as a support vector machine, an adaptive boosting classifier, a neural network classifier, and so on. A support vector machine operates by finding a hyper-surface in the space of possible inputs. The hyper-surface attempts to split the positive examples from the negative examples by maximizing the distance between the nearest of the positive and negative examples to the hyper-surface. This allows for correct classification of data that is similar to but not identical to the training data. Various techniques can be used to train a support vector machine. One technique uses a sequential minimal optimization algorithm that breaks the large quadratic programming problem down into a series of small quadratic programming problems that can be solved analytically. (*See* Sequential Minimal Optimization, at http://research.microsoft.com/~jplatt/smo.html.)

[0025] Adaptive boosting is an iterative process that runs multiple tests on a collection of training data. Adaptive boosting transforms a weak learning algorithm (an algorithm that performs at a level only slightly better than chance) into a strong learning algorithm (an algorithm that displays a low error rate). The weak learning algorithm is run on different subsets of the training data. The algorithm concentrates more and more on those examples in which its predecessors tended to show mistakes. The algorithm corrects the errors made by earlier weak learners. The algorithm is adaptive because it adjusts to the error rates of its predecessors. Adaptive boosting combines rough and moderately inaccurate rules of thumb to create a high-performance algorithm. Adaptive boosting combines the results of each separately run test into a single, very accurate classifier.

[0026] A neural network model has three major components: architecture, cost function, and search algorithm. The architecture defines the functional form relating the inputs to the outputs (in terms of network topology, unit connectivity, and activation functions). The search in weight space for a set of weights that minimizes the objective function is the training process. A neural network model may use a radial basis function ("RBF") network and a standard gradient descent as its search technique.

[0027] Figure 1 is a diagram that illustrates a graph of documents returned as the search result of a query. In this example, subgraph 100 represents a portion of the documents returned in the search result. Nodes 101-112 represent 12 documents of the search result. Nodes 101 and 106 represent labeled documents. The document represented by node 101 has been labeled with the relevance score of .75, and the document represented by node 106 has been labeled with the relevance score of .6. The propagation system generated the edges between the nodes using a nearest neighbor algorithm. In this example, nodes 102, 103, and 104 are each one of the k-nearest neighbors to node 101, but nodes 105-112 are not one of the k-nearest neighbors. The propagation system then calculated the similarity between connected nodes using a similarity score algorithm. For example, node 101 is connected to node 102 with an edge with the weight of .8, which indicates the similarity between the connected nodes.

[0028] Figure 2 is a block diagram that illustrates components of the propagation system in one embodiment. The propagation system 230 is connected to document stores 210 (e.g., web sites) via communications link 220 (e.g., Internet). The propagation system includes a collect training data component 231, a training data store 232, and a document index 233. The document index contains an index of documents (e.g., web pages) in the document stores. The document index may be generated by a web crawler. The document index may include a feature vector for each document that is used for training a ranking function. The feature vectors may represent many different types of features of documents such as inverse document frequency, keywords, font size, and so on. The collect training data component submits queries to a search engine (not shown), and receives documents that match the queries. The search engine may be independent of the propagation system. In such a case, the propagation system may generate feature vectors dynamically from search results. The collect training data component may prompt a user to label the relevance of some of the documents that match the queries. The collect training data component stores the queries, search results (e.g., feature vectors), and labels in the training data store. The propagation system also includes a propagate relevance component 235, a build graph component 236, a generate weights for graph component 237, a normalize weights of graph component 238, and a propagate

relevance based on graph component 239. The propagate relevance component propagates the relevance of the labeled documents to the unlabeled documents that are stored in the training data store. The propagate relevance component invokes the build graph component to build a graph including edges representing the documents of a search result. The propagate relevance component then invokes the generate weights for graph component to generate the initial weights for the edges of the graph. The propagate relevance component invokes the normalize weights of graph component to normalize the generated weights. The propagate relevance component then invokes the propagate relevance based on graph component to perform the actual propagation of the relevance from the labeled documents to the unlabeled documents. The propagation system also includes a create ranking function component 241 and a ranking function 242. The create ranking function uses the training data with the propagated relevance to create a ranking function.

[0029] The computing devices on which the propagation system may be implemented may include a central processing unit, memory, input devices (e.g., keyboard and pointing devices), output devices (e.g., display devices), and storage devices (e.g., disk drives). The memory and storage devices are computer-readable media that may contain instructions that implement the propagation system. In addition, the data structures and message structures may be stored or transmitted via a data transmission medium, such as a signal on a communications link. Various communications links may be used, such as the Internet, a local area network, a wide area network, or a point-to-point dial-up connection.

[0030] The propagation system may provide services to various computing systems or devices including personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0031] The propagation system may be described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs,

objects, components, data structures, and so on that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0032] Figure 3 is a flow diagram that illustrates the processing of the create ranking function component of the propagation system in one embodiment. The create ranking function component collects training data, propagates the relevance of labeled documents to unlabeled documents, and then trains a ranking function. In block 301, the component collects the training data. In block 302, the component inputs labels for a subset of the training data. In block 303, the component invokes the propagate relevance component to propagate the relevance of the labeled documents to the unlabeled documents. In block 304, the component trains the ranking function using the propagated relevances.

[0033] Figure 4 is a flow diagram that illustrates the processing of the propagate relevance component of the propagation system in one embodiment. The component is provided training data and propagates the relevance of the labeled documents to the unlabeled documents. In block 401, the component invokes the build graph component to build the initial graph that includes edges. In block 402, the component invokes the generate weights for graph component to generate weights indicating the similarity between documents represented by connected nodes. In block 403, the component invokes the normalize weights of graph component to normalize the weights of the graph. In block 404, the component invokes the propagate relevance based on graph component to perform the propagation of the relevance. The component then returns.

[0034] Figure 5 is a flow diagram that illustrates the processing of the build graph component of the propagation system in one embodiment. The component creates a square matrix with each row and column representing a document. The component then identifies and adds a connection between each node and its k-nearest neighbors (e.g., k=10). In block 501, the component selects the next document i. In decision block 502, if all the documents i have already been selected, then the component returns, else the component continues at block 503. In block 503, the component selects the next document j. In decision block 504, if all the documents j for the

selected document i have already been selected, then the component continues at block 506, else the component continues at block 505. In block 505, the component calculates the distance between the selected document i and the selected document j and then loops to block 503 to select the next document j. In block 506, the component selects the 10 documents j with the smallest distance for a document i (i.e., the nearest neighbors) and then loops to block 501 to select the next document i.

[0035] Figure 6 is a flow diagram that illustrates the processing of the generate weights for graph component of the propagation system in one embodiment. The component calculates the similarity between connected documents based on a Manhattan metric. In block 601, the component selects the next document i. In decision block 602, if all the documents i have already been selected, then the component returns, else the component continues at block 603. In block 603, the component initializes the similarity of the document to itself to 0. In block 604, the component selects the next nearest document j (i.e., a connected document) to the selected document i. In decision block 605, if all the nearest documents j to the selected document i have already been selected, then the component loops to block 601 to select the next document i, else the component continues at block 606. In block 606, the component initializes the similarity between the selected document i and the selected document j to 1. In blocks 607-609, the component loops calculating the distance metric. In block 607, the component selects the next dimension l of the feature vector. In decision block 608, if all the dimensions have already been selected, then the component loops to block 604 to select the next nearest document j, else the component continues at block 609. In block 609, the component sets the similarity between the selected document i and the selected document j to its current similarity multiplied by a function of the difference between the selected feature l of the selected document i and the selected document j in accordance with Equation 2. The component then loops to block 607 to select the next dimension.

[0036] Figure 7 is a flow diagram that illustrates the processing of the normalize weights of graph component of the propagation system in one embodiment. The

component normalizes the weights of the similarity matrix. In block 701, the component selects the next row i of the similarity matrix. In decision block 702, if all the rows have already been selected, then the component continues at block 706, else the component continues at block 703. In blocks 703-705, the component calculates the value for the diagonal matrix D for the selected row. In block 703 the component selects the next column j of the similarity matrix. In decision block 704, if all the columns have already been selected, then the component loops to block 701 to select the next row, else the component continues at block 705. In block 705, the component adds the weights of the selected row i and the selected column j to the diagonal element for the selected row i. The component then loops to block 703 to select the next column j for the selected row i. In block 706, the component normalizes the similarity matrix in accordance with Equation 3.

[0037] Figure 8 is a flow diagram that illustrates the processing of the propagate relevance based on graph component of the propagation system in one embodiment. The component iteratively calculates the Taylor series expansion of Equation 5 until it converges on a solution. In block 801, the component initializes index i to zero. In block 802, the component initializes the solution vector to 0. In blocks 803-805, the component loops until it converges on a solution. In block 803, the component calculates the value for the next iteration based on the value of the previous iteration plus the next factor of the Taylor series expansion. In decision block 804, if the values converge on a solution, then the component returns, else the component continues at block 805. In block 805, the component increments the index to the next iteration and loops to block 803 to perform the next iteration.

[0038] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims. The propagation system may be used to augment search results. For example, a search engine may generate a search result based on a certain corpus of documents. The relevance of the documents of the search result can then be propagated to documents of a different

corpus using the propagation system. The documents of the different corpus with the highest relevance can then be added to the search result. The propagation system may be used to propagate relevance from documents labeled with their relevance to a single query to unlabeled documents (intra-query propagation) or from documents labeled with their relevance to multiple queries to unlabeled documents (inter-query propagation). The propagation component trains the training component separately for each query with intra-query propagation and simultaneously for multiple queries with inter-query propagation. Accordingly, the invention is not limited except as by the appended claims.

CLAIMS

I/We claim:

[c1] 1. A system for propagating relevance of labeled documents to unlabeled documents, comprising:

a document store (232) that contains representations of documents, some of the documents being labeled with relevance to a query and others of the documents not being labeled with relevance to the query;

a graph component (236) that creates a graph of the documents with the documents represented as nodes being connected by edges representing similarity between documents; and

a propagate relevance component (239) that propagates relevance of the labeled documents to the unlabeled documents based on the similarity between documents as indicated by the similarity represented by the edges of the graph.

[c2] 2. The system of claim 1 wherein the graph component includes:

a build graph component that builds a graph in which nodes representing similar documents are connected via edges;

a generate weights component that generates weights for the edges based on similarity of the documents represented by the connected nodes; and

a normalize weights component that normalizes the weights of the graph.

[c3] 3. The system of claim 2 wherein the build graph component establishes edges between nodes using a nearest neighbor algorithm.

[c4] 4. The system of claim 3 wherein the nearest neighbor algorithm uses a Euclidean distance metric.

[c5] 5. The system of claim 3 wherein the build graph component connects a node to its 10 nearest neighbors.

[c6] 6. The system of claim 2 wherein the build graph component establishes edges between each pair of nodes.

[c7] 7. The system of claim 2 wherein the build graph component establishes edges between nodes to create a minimum spanning tree.

[c8] 8. The system of claim 1 wherein the relevance of the labeled documents is generated by searching for documents related to the query in a corpus of documents and the unlabeled documents are not included in the corpus of documents.

[c9] 9. The system of claim 1 wherein the propagate relevance component propagates relevance using a manifold ranking based algorithm.

[c10] 10. The system of claim 1 wherein the propagate relevance component propagates relevance according to the following equation:

$$f^* = (1-\alpha)(I-\alpha S)^{-1} y$$

where f^* represents a propagated relevance vector, S is a similarity matrix, y represents an initial relevance vector, and α represents a decay rate.

[c11] 11. The system of claim 1 wherein the propagate relevance component propagates relevance according to the following equation:

$$f^* = \left(I+\alpha S+\alpha^2 S^2+\ldots+\alpha^N S^N\right) y$$

where f^* represents a propagated relevance vector, S is a similarity matrix, y represents an initial relevance vector, and α represents a decay rate, and where n represents an exponent for which f^* converges on a solution.

[c12] 12. A system for propagating relevance of labeled pages to a query to unlabeled pages to the query, comprising:

a page store (232) that contains representations of pages, some of the pages being labeled with relevance to a query and others of the pages not being labeled with relevance to the query;

a graph component that creates a graph of the pages with the pages represented as nodes connected by edges representing similarity between the pages, including:

a build graph component (236) that builds a graph in which nodes representing similar pages are connected via edges; and

a generate weights component (237) that generates weights for the edges based on similarity of the pages represented by the connected nodes; and

a propagate relevance component (239) that propagates relevance of the labeled pages to the unlabeled pages based on the similarity between pages as indicated by the similarity represented by the edges of the graph and based on a manifold ranking algorithm.

[c13] 13. The system of claim 12 wherein the build graph component establishes edges between nodes using a nearest neighbor algorithm.

[c14] 14. The system of claim 13 wherein the nearest neighbor algorithm uses a Euclidean distance metric.

[c15] 15. The system of claim 13 wherein the build graph component connects a node to its 10 nearest neighbors.

[c16] 16. The system of claim 12 wherein the generate weights component uses a Manhattan distance metric for representing the similarity between pages.

[c17] 17. The system of claim 12 wherein each page is represented by a feature vector and the similarity between pages is represented by distance in feature vector space.

[c18] 18. A computer-readable medium containing instructions for controlling a computer system to propagate relevance of documents to a query to other documents, by a method comprising:

creating (236) a graph of the documents represented as nodes connected by edges having weights representing similarity between documents; and

propagating (239) the relevance of the labeled documents to the unlabeled documents based on the weights of the edges between nodes using a manifold ranking based algorithm.

[c19] 19. The computer-readable medium of claim 18 wherein the propagating of the relevance of the labeled documents includes using a Taylor expansion to iteratively solve the following equation:

$$f^* = (1-\alpha)(I-\alpha S)^{-1} y$$

[c20] 20. The computer-readable medium of claim 18 wherein the creating of the graph includes connecting edges using a nearest neighbor algorithm and establishing the weight of an edge based on distance between documents represented by the nodes connected by the edge.