TOWARD A QUALITATIVE SEARCH ENGINE

Traditional search engines do not consider document quality in ranking search results. The Hyperlink Vector Voting method adds a qualitative dimension to its rankings by factoring in the number and descriptions of hyperlinks to the document.

YANHONG LI
GARI Software / IDD Information Services

Hypertext is fundamentally a database system that provides a unique, nonsequential method of accessing information. Its essential features are nodes and hyperlinks. Nodes contain text, graphics, audio, video, and other media. Hyperlinks connect the nodes and support the nonlinear organization of information. The most extensive and popular hypertext system today is the World Wide Web. However, unlike a traditional document database, where each document is selected or authored for its relevance to the whole, the Web exists in the open environment of the Internet where everyone can post documents. The fact that a resource exists on the Internet is no guarantee of its importance, accuracy, utility, or value.[1]

Most Internet searches are launched by casual users making simple queries on popular topics.[2] Because such queries can generate tens of thousands of hits, the ranking of a document's relevance to a query is a core technology for search engines. Traditional information retrieval theory offers models for measuring the similarity between user-defined keywords and document contents. These models include the Vector Space Model, probabilistic models, and fuzzy logic models.[3] Almost all models depend on the frequency of query terms in a given document (there are ways to normalize the weighting of terms to account for document length[4]). However, this approach assumes an integrity among the documents in a database that is not at all the case on the Internet. On the contrary, "keyword spamming" is a common technique used by Web-site marketers to increase their ranking in search results. In any case, the frequency with which a word appears in a document is no guarantee of content quality.

This article describes the Hyperlink Vector Voting method, a new method of indexing and retrieving hypertext documents. HVV uses the content of hyperlinks to a document to rank its relevance to the query terms. Thus, HVV operates somewhat like the Science Citation Index, where the number of papers that cite a given paper has proved an effective tool for measuring the quality of its content and its relevance to a topic. In HVV, the hyperlinks to a site are the "citations" to it.

After introducing some basic information retrieval concepts and their relation to a hypertext implementation, I describe the HVV method in some detail, followed by experimental results for Rankdex, a simple search engine I developed that uses the HVV method.

INFORMATION RETRIEVAL

In an information retrieval system, documents are preprocessed (inverted) to create an *inverted index* that records for each term its *postings* in the collection. A posting is a tuple consisting of a *term*, a *document id* (a unique document identifier, typically an integer), and the weight of that term in that document. The postings associated with a term are sorted by document id. Similarity-based search algorithms consult the postings of each query term to compute the relevance scores of documents that have terms in common with the query.

The Vector Space Model[5] is widely used in information retrieval to quantify the similarity between two documents, or a query and a document. In this model, a document is represented by a vector with one component for each unique term in the vocabulary of the collection. The value of each component is the weight for that term in that document, often a function of the frequency of the term in that document. Terms that do not occur in the document have zero weight. Queries are also represented as vectors. The similarity between a query and a document is then calculated as the inner-product of their term vectors. This measure is equal to the cosine of the angle between the two vectors, and hence is sometimes called "cosine similarity."

For example, in the vector-space model, the user query $\vec{Q}$ is a vector and each keyword in the query represents a dimension of the query vector; m is the number of keywords in the query. For Internet search engines, 70 percent of all queries have an m value of one or two, but m could be very large in some cases.

$$\vec{Q} = (q_1, q_2, \dots, q_m)$$

A document, $\vec{D}$, is also represented by a vector, with each keyword as a dimension, where n is typically the number of keywords in $\vec{D}$.

$$\vec{D} = (d_1, d_2, \dots, d_m)$$

The relevance score is then calculated by the dot product of $\vec{Q}$ and $\vec{D}$ after $\vec{Q}$ and $\vec{D}$ are normalized to have the same dimension. (If a keyword appearing in $\vec{D}$ does not belong to $\vec{Q}$, the corresponding dimension will have a value of 0 in $\vec{Q}$ and vice versa.)

$$R = \vec{Q} \cdot \vec{D} \qquad (1)$$

The calculation of a dimension's value in $\vec{Q}$ or $\vec{D}$ is called *term weighting*. The most popular term weighting formula is

$$W_{x,t} = f_{x,t} \cdot \log(N / f_t) \qquad (2)$$

where $f_{x,t}$ is the number of occurrences of word t in x, which could be either a query or document; N is the number of documents in the collection; and f_t is the number of documents containing term t.

This function allots high weights to rare words on the assumption that these words are more discriminating than common words; in other words, the presence of a rare word in both a query and a document is assumed to be a good relevance indicator.

This assumption is not valid in a hypertext system like the Web where the documents have highly variable qualities and purposes. However, hypertext systems do offer another option for determining relevancy in that hypertext documents express not only their semantic content but also their hyperlinks to other documents (see the sidebar, "Hyperlink Structure").

IR research in the context of hypertext has therefore focused on the user-specified links between nodes. This research includes efforts to integrate

HYPERLINK STRUCTURE

A *hyperlink* connects two anchors on different nodes: the hyperlink's head (at the destination node) and its tail (at the source). An *anchor* is identified by an absolute Uniform Resource Identifier, such as http://www.w3.org/hypertext/Book.html. A fragment identifier can also be added, indicated by a "#" and a sequence of characters: http://www.w3.org/hypertext/Book.html#chapter1.

In HTML, hyperlinks are formatted with different options:

<a option1 , . . . , optionN> anchor-text </a>.

The most important option is HREF, which defines a hyperlink: <a href="URL"> anchor-text </a>, where the URL is a Uniform Resource Locator for the head (destination) anchor and "anchor-text" is the tail (source) anchor, which typically describes the document specified by the URL.

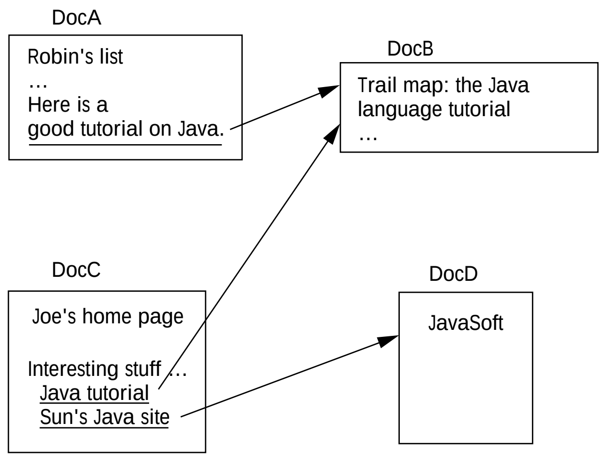

Figure 1. A simple hypertext system. In DocA, the hyperlink's head anchor is the URL to DocB; the tail anchor (and hyperlink content) is "good tutorial on Java."

query-based retrieval strategies with browsing in hypertext networks[6,7] and experiments to improve retrieval performance using hyperlinks and connected nodes.[8] There has also been work on improving retrieval effectiveness by combining the evidence represented by different types of links in a network representation.[9]

These hypertext projects, however, follow traditional IR in ranking the retrieval result primarily according to document content. They consider hyperlinks as additional information and usually give them lower weight in relevance ranking. Furthermore, hypertext IR research usually factors in the content of documents at connected nodes in its ranking. These additional documents complicate the ranking to the point where it is ineffective in general, even though it can be very effective in specific situations. For example, using connected nodes for additional information can improve retrieval quality when the collection is small and its documents belong to a specific field, such as medicine or computer science.

THE HYPERLINK VECTOR VOTING METHOD

The HVV method ranks a document on the basis of the number of hyperlinks to it and uses the hyperlink anchor-text as the semantic content for the document. In essence, when the database is large enough, the search results are similar to "voting" results: a document's content is selected based on how others describe it, rather than how the author describes it. The HVV method uses link vectors to represent a document. Each link vector represents a hyperlink pointing to the document; in other words, the hyperlink's head anchor is the document's URL. The hyperlink's tail—its anchor-text—is treated as its content, and the weighting of each term in the anchor-text defines one link vector for the document.

During indexing, the index engine traverses the whole document database using a Spider, collects hyperlink information for each document, and creates a list in the form

Doc.ID: anchor-text

where Doc.ID refers to a hyperlink's head anchor. The search engine can generate an inverted index from this information, typically in the form

Term: DF, Doc_1, Doc_2, … , Doc_i, … , Doc_{DF}

where Term is a term from the anchor-text, DF is Term's document frequency (defined as the number of documents that are referred as the head anchor in any hyperlink with anchor-text that contains Term), and Doc_i is the ith document hyperlinked to by anchor texts with Term.

Finally, a hypertext document is represented by link vectors

$$D_j = \begin{bmatrix} \vec{L}_1 \\ \vec{L}_2 \\ \cdots \\ \vec{L}_i \end{bmatrix} \qquad (3)$$

where D_j is a document ID, and $\vec{L}_i$ is the ith hyperlink vector whose head anchor is D_j. The value of each link-vector dimension is calculated using a term weighting method.

Figure 1 shows a small hypertext system with four nodes (DocA, DocB, DocC, and DocD) and three hyperlinks (from DocA to DocB, DocC to DocB, and DocC to DocD). During the traversing process, the link information is extracted as

DocB: "good tutorial on Java"; "Java Tutorial"
DocD: "Sun's Java Site"

The inverted file produced from the link information will be (assume it is case insensitive):

good	1	DocB
tutorial	1	DocB
on	1	DocB

java	2	DocB, DocD
sun's	1	DocD
site	1	DocD

and the hyperlink vector representation for the documents are:

	good tutorial on java
DocB:	< 2, 2, 2, 1 >
	java tutorial
	< 1, 2 >
	Sun's java site
DocD:	< 2, 1, 2 >

As with most search engines, during information retrieval, the HVV method matches query words against the inverted index to locate relevant documents. HVV then ranks the documents based on the hyperlink vector representations. Like link descriptions, queries are also represented as vectors. The ranking score is defined as the sum of all dot products between the query vector and each hyperlink vector for a given document. The summation process is like voting: more links will typically result in higher scores. However, the voting is weighted; it also depends on similarities between the link and query vectors. Equation 4 shows the ranking formula.

$$R = \sum_{i=1}^{n} \left(\vec{Q} \cdot \vec{L}_i \right) \tag{4}$$

where R is the ranking score, $\vec{Q}$ is the query vector, and $\vec{L}_i$ is the link vector. For example, in the hypertext system in Figure 1, if the query is "Java Tutorial," it is represented as vector

java tutorial
Query: < 1, 2 >

Similarly, we can calculate the document link vectors for DocB: ranking scores for DocB and DocD are 1.62 and 0.149, respectively.

Because no hyperlinks point to document A or C, their ranking scores for the query "Java tutorial" are zero, even though both documents contain the words "Java" and "tutorial." When HVV assigns the same rating to two documents, conventional indexing and retrieval methods can be used for further distinction. In this case, conventional relevance scores for documents A and C would be 0.6 and 0.8, respectively, giving a final relevance ranking of DocB, DocD, DocC, and DocA.

EXPERIMENTAL RESULTS

In 1997, we conducted tests using Rankdex,[10] an experimental HVV-based Web search engine available at **http://rankdex.gari.com**. Our spider collected about 5.3 million hypertext documents on the Internet and indexed them according to the HVV index structure. The total size of the index is 100 Mbytes (much smaller than a conventional index, which is typically around 500 Mbytes).

Because we do not have a truth value for a query set for the Web documents, we conducted experiments in an ad hoc way. In general, documents reviewed by people and judged as relevant are accepted as close to the truth value of a user query. Therefore, for our experiments we collected a set of popular queries and sent them to a search engine that manually reviews and categorizes Web documents, what we call the "editor's search engine." We sent the same queries to Rankdex and counted how many of Rankdex's top 10 hits were returned by the editor's search engine and selected as relevant. For comparison, we sent the queries to other major search engines and counted how many of their top 10 hits were selected by the editor's search engine.

Because the average number of keywords specified in user queries is 1.5,[11] we used a small but representative set of 10 short search queries. Finally, many search engines have their own team of editors and collection of editors' choices. Because we were comparing Rankdex with many major search

Table 1. Search results comparison.

Query	LookSmart	Rankdex	AltaVista	Excite	Infoseek	Lycos
Wallpaper	10	2	0	1	0	0
Skiing	10	2	0	1	0	0
Real estate	10	1	0	1	0	0
Sandra Bullock	5	3	0	0	2	0
Yahoo	8	2	1	0	1	0
Microsoft	10	1	0	0	0	0
Las Vegas	9	1	0	0	0	0
Airlines	4	2	0	1	0	0
Stock quotes	10	2	1	0	0	0
Java tutorial	3	2	0	1	1	0
TOTAL	79	18	2	5	4	0

Table 2. Excite-HVV search results comparison.

Query	LookSmart	Rankdex	Excite	Excite-HVV
Wallpaper	10	2	1	1
Skiing	10	2	1	2
Real estate	10	1	1	1
Sandra Bullock	5	3	0	3
Yahoo	8	2	1	1
Microsoft	10	1	0	1
Las Vegas	9	1	0	1
Airlines	4	2	1	2
Stock quotes	10	2	0	1
Java tutorial	3	2	1	2
TOTAL	79	18	6	15

engines, for our editor's engine we chose one that was lesser known but well regarded: LookSmart (**http://www.looksmart.com**) from Reader's Digest.

Table 1 shows the search results. We used 10 queries on six search engines, examining the top 10 results for each. LookSmart editors chose 79 Web sites for the 10 queries. Rankdex matched 18 of LookSmart's search results, roughly 22.7 percent. The closest commercial search engine, Excite, matched five (6.3 percent). Of the remaining engines, Infoseek matched four (5.0 percent), AltaVista matched one (1.2 percent), and Lycos matched none. HVV most closely simulated human editors' efforts and could therefore presumably meet users' information needs better than the other search engines evaluated.

Not all of the 79 documents returned by LookSmart were completely relevant. For example, for the query "real estate," the home page for Arthur Andersen (a consulting company) was among the top 10. Clearly, it is not a good match. On the other hand, Rankdex and other search engines did return some good matches not reviewed or returned by LookSmart. Thus, the low matching percentage does not mean the search engines are much worse than LookSmart. Nevertheless, all of the overlapped documents are relevant to the queries and of good quality.

Because the Rankdex index was significantly smaller than the other search engine indexes—5.3 million pages as compared with 20–50 million—we did another test to verify that HVV benefits also apply to larger hypertext systems. In our "meta-engine test," we took the first *N* search results from a major search engine and applied HVV indexing and retrieval: We indexed the *N* pages, extracted the hyperlinks and anchor texts, and computed the relevance scores using the HVV method. We then reranked the *N* documents and compared the overlap with the new top-10 list and the LookSmart search results.

As a base search engine, we used Excite with *N* = 500. Table 2 shows the results, using the same query set as before. The Excite-HVV column shows the overlap with LookSmart when HVV is applied to the first 500 hits from an Excite query. For comparison, Rankdex and Excite alone are also listed.

Excite-HVV achieved a total of 15 overlaps, a 300 percent increase over Excite alone. This is still less than Rankdex alone, possibly because only the top 500 hits of Excite were counted. If *N* is large enough, we might see even better results: the larger the database, the more the votes, and hence the more objective the results.

BENEFITS AND DRAWBACKS OF HVV

Because the hyperlink vector and link-based inverted index contain only link information, ranking does not depend on the words appearing in the documents themselves. Rather, rankings are based only on hyperlinks—how many there are to a given document and the document description the links provide. This solves several problems common to traditional ranking systems. Documents that use "keyword spamming" will rank high only if they meet the description and popularity standard as well. Document size is no longer a factor in relevance ranking and thus shorter documents that may be more relevant but do not use a term as often as a long document are more likely to be selected. Thesaurus or knowledge bases might also be less important. For example, even if the word "lawyer" never appeared in a document titled "California Immigration Attorneys," the word might be used in a hyperlink pointing to this document.

There are other advantages as well. Images, graphics, and sounds—which are not searchable by conventional methods—are searchable by hyperlink descriptions pointing to them. The same is true of foreign-language documents if there are hyperlinks to them in the user's native language. In cases where images, graphics, and so on serve as anchor text, the index engine simply substitutes an applicable image or graphic with the tail anchor's document title.

The HVV model can also derive applications, such as one that automatically selects the "Best of the Web" in any particular area. Also, by comparing different descriptions of hyperlinks pointing to the same document, you can discover synonyms, extract new concepts, and build a thesaurus.

A potential problem is that HVV-based engines can also be spammed. In experiments so far, we have counted duplicate hyperlinks and link descriptions on the same Web site as multiple links. Given this, people could construct documents whose contents are nothing but repeated links to get a high rank for the links' head document. However, the potential for "link spamming" is easily thwarted: we simply count each link only once, no matter how many times it appears on a Web site. Getting multiple Web sites to include links to a certain site with the intent to spam is clearly a more difficult project.

FUTURE WORK

We are currently investigating many issues, including when best to use HVV in place of traditional relevance ranking. Because HVV is kind of "voting," if there are not many query-related documents, voting results might be random. Even in a large hypertext system such as the Web, HVV would probably work better in combination with traditional retrieval methods. Whenever the confidence score of HVV is low, traditional relevance ranking should be used.

Because of the dynamic nature of the Web, hyperlinks often point to bad URLs. Therefore, detecting bad links is a practical next step. Also, because different URLs might point to the same document, detecting duplicate URLs and combining them into a single document ID is a challenging task.

Given the different ways HVV and traditional IR methods define document frequency, it would be interesting to compare the actual distribution of words on the Internet according to each method. Finally, HVV is derived from the Vector Space Model; it would be interesting to see how the spirit of hyperlink indexing could be adopted by probabilistic and other retrieval models. ■

ACKNOWLEDGMENTS

I thank Larry Rafsky for his strong support of this research, Doran Howitt for his effort in deploying a real-world search service based on Rankdex, and Ron Katriel for his constructive suggestions. Many other members of GARI Software provided important feedback after using Rankdex. I also thank William Chang of Infoseek for beneficial discussions and the anonymous reviewers for their suggestions. This research is funded by IDD Information Services.

REFERENCES

1. H. Berghel, "Cyberspace 2000: Dealing with Information Overload," *Comm. ACM*, Vol. 40, No. 2, 1997, pp. 19-24.
2. E. Selberg and O. Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler," *Proc. Fourth Int'l WWW Conf.*, 1995; available online at http://www.w3.org/Conferences/WWW4/Papers/169.
3. D. Harman, "Ranking Algorithms," in *Information Retrieval Data Structures and Algorithms*, W.B Frakes and R. Baeza-Yates, eds., Prentice Hall, Upper Saddle River, N.J., 1992, pp. 363-392.
4. G. Singhal, A. Salton, and C. Buckley, "Length Normalization in Degraded Text," *Fifth Symp. Document Analysis and Information Retrieval*, 1996; available online at http://www.research.att.com/~singhal/ocr-norm.ps.
5. G. Salton, *The SMART Retrieval System*, Prentice-Hall, Upper Saddle River, N.J., N..J., 1971.
6. M.E. Frisse, "Searching for Information in a Hypertext Medical Handbook," *Comm. ACM*, Vol. 31, No. 7, 1988, pp. 880-886.
7. R. Thompson and W.B. Croft, "Support for Browsing in an Intelligent Text Retrieval System," *Int'l J. Man-Machine Studies*, Vol. 30, No. 6, 1989, pp.639-668.
8. H.P. Frei and D. Stieger, "The Use of Semantic Links in Hypertext Information Retrieval," *Information Processing & Management*, Vol. 31, No. 1, 1995, pp. 1-13.
9. W.B. Croft and H. Turtle, "A Retrieval Model for Incorporating Hypertext Links," *Hypertext 89 Proc.*, ACM Press, Pittsburgh, 1989, pp. 213-224.
10. Y. Li, "Beyond Relevance Ranking: Hyperlink Vector Voting," *RIAO 97: Computer-Assisted Information Searching on Internet*, 1997, McGill University, Montreal, Canada, pp. 648–650. Also available at http://ciir.cs.umass.edu/nir97/li.ps.gz.
11. B. Pinkerton, "Finding What People Want: Experiences with the WebCrawler," *Proc. Second Int'l WWW Conf.*, 1994; available online at http://info.webcrawler.com/bp/www94.html.

Yanhong Li is a staff software engineer for Infoseek Corporation's core technology group. His research interests include Internet search engines, information retrieval, document analysis, and new media management. From 1994 to 1997, he was a senior consultant at GARI Software, IDD Information Services. He received a BS degree in information science from Peking University, China, in 1991 and an MS degree in computer science from the State University of New York at Buffalo in 1993.

Readers may contact the author at Infoseek Corporation, 1399 Moffett Park Drive, Sunnyvale, CA 94089; yanhong@infoseek.com.