Information Retrieval and Web Search

Amy N. Langville* Carl D. Meyer†

January, 2006

Information retrieval is the process of searching within a document collection for information most relevant to a user's query. However, the type of document collection significantly affects the methods and algorithms used to process queries. In this chapter we distinguish between two types of document collections: traditional and Web collections. Traditional information retrieval is search within small, controlled, nonlinked collections (e.g., a collection of medical or legal documents), whereas Web information retrieval is search within the world's largest and linked document collection. In spite of the proliferation of the Web, more traditional nonlinked collections still exist, and there is still a place for the older methods of information retrieval.

1 The Traditional Vector Space Method

Today most search systems that deal with traditional document collections use some form of the vector space method [SB83] developed by Gerard Salton in the early 1960s. Salton's method transforms textual data into numeric vectors and matrices and employs matrix analysis techniques to discover key features and connections in the document collection.

Definitions:

For a given collection of documents and for a dictionary of m terms, document i is represented by an $m \times 1$ **document vector** $\mathbf{d}_i$ whose jth element is the number of times term j appears in document i.

*Department of Mathematics, The College of Charleston, Charleston, SC 29424 USA,

†Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA, This work was supported in part by the National Science Foundation under NSF grant CCR-0318575.

The **term-by-document matrix** is the $m \times n$ matrix

$$A = [\, \mathbf{d}_1 \, \mathbf{d}_2 \, \cdots \, \mathbf{d}_n \,]$$

whose columns are the document vectors.

Recall is a measure of performance that is defined to be

$$0 \leq \text{Recall} = \frac{\#\text{ relevant docs retrieved}}{\#\text{ relevant docs in collection}} \leq 1.$$

Precision is another measure of performance defined to be

$$0 \leq \text{Precision} = \frac{\#\text{ relevant docs retrieved}}{\#\text{ docs retrieved}} \leq 1.$$

Query processing is the act of retrieving documents from the collection that are most related to a user's query, and the **query vector** $\mathbf{q}_{m\times 1}$ is the binary vector defined by

$$q_i = \begin{cases} 1 & \text{if term } i \text{ is present in the user's query,} \\ 0 & \text{otherwise.} \end{cases}$$

The **relevance** of document i to a query $\mathbf{q}$ is defined to be

$$\delta_i = cos\, \theta_i = \mathbf{q}^T \mathbf{d}_i / \|\mathbf{q}\|_2 \|\mathbf{d}_i\|_2.$$

For a selected tolerance τ, the **retrieved documents** that are returned to the user are the documents for which $\delta_i > \tau$.

Facts:

1. The term-by-document matrix A is sparse and nonnegative, but otherwise unstructured.
2. [BB05] In practice, weighting schemes other than raw frequency counts are used to construct the term-by-document matrix because weighted frequencies can improve performance.
3. [BB05] Query weighting may also implemented in practice.
4. The tolerance τ is usually tuned to the specific nature of the underlying document collection.
5. Tuning can be accomplished with the technique of relevance feedback, which uses a revised query vector such as $\tilde{\mathbf{q}} = \delta_1 \mathbf{d}_1 + \delta_3 \mathbf{d}_3 + \delta_7 \mathbf{d}_7$, where $\mathbf{d}_1$, $\mathbf{d}_3$, and $\mathbf{d}_7$ are the documents the user judges most relevant to a given query $\mathbf{q}$.
6. When the columns of A and $\mathbf{q}$ are normalized, as they usually are, the vector $\boldsymbol{\delta}^T = \mathbf{q}^T A$ provides the complete picture of how well each document in the collection matches the query.

7. The vector space model is efficient, because A is usually very sparse, and $\mathbf{q}^T A$ can be executed in parallel, if necessary.

8. [BB05] Because of linguistic issues such as polysomes and synonyms, the vector space model provides only decent performance on query processing tasks.

9. The underlying basis for the vector space model is the standard basis $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m$, and the orthogonality of this basis, can impose an unrealistic independence among terms.

10. The vector space model is good starting place, but variations have been developed that provide better performance.

Examples

1. Consider a collection of 7 documents and 9 terms (taken from [BB05]). Terms not in the system's index are ignored. Suppose further that only the titles of each document are used for indexing. The indexed terms and titles of documents are shown below.

Terms	Documents
T1: Bab(y,ies,y's)	D1: *Infant* & *Toddler* First Aid
T2: Child(ren's)	D2: *Babies* and *Children's* Room (For Your *Home*)
T3: Guide	D3: *Child Safety* at *Home*
T4: Health	D4: Your *Baby's Health* and *Safety*: From *Infant* to *Toddler*
T5: Home	D5: *Baby Proofing* Basics
T6: Infant	D6: Your *Guide* to Easy Rust *Proofing*
T7: Proofing	D7: Beanie *Babies* Collector's *Guide*
T8: Safety	
T9: Toddler	

The indexed terms are italicized in the titles. Also, the stems [BB05] of the terms for baby (and its variants) and child (and its variants) are used to save storage and improve performance. The term-by-document matrix for this document collection is

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

For a query on *baby health*, the query vector is

$$\mathbf{q} = [\,1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\,]^T .$$

To process the user's query, the cosines

$$\delta_i = cos\,\theta_i = \frac{\mathbf{q}^T\mathbf{d}_i}{\|\mathbf{q}\|_2\|\mathbf{d}_i\|_2}$$

are computed. The documents corresponding to the largest elements of $\boldsymbol{\delta}$ are most relevant to the user's query. For our example,

$$\boldsymbol{\delta} \approx [\,0 \quad 0.40824 \quad 0 \quad 0.63245 \quad 0.5 \quad 0 \quad 0.5\,],$$

so document vector 4 is scored most relevant to the query on *baby health*. To calculate the recall and precision scores, one needs to be working with a small, well-studied document collection. In this example documents $\mathbf{d}_4$, $\mathbf{d}_1$ and $\mathbf{d}_3$ are the three documents in the collection relevant to baby health. Consequently, with $\tau = .1$, the recall score is $1/3$ and the precision is $1/4$.

2 Latent Semantic Indexing

In the 1990s, an improved information retrieval system replaced the vector space model. This system is called Latent Semantic Indexing (LSI) [Dum91] and was the product of Susan Dumais, then at Bell Labs. LSI simply creates a low rank approximation A_k to the term-by-document matrix A from the vector space model.

Facts:

1. [Mey00] If the term-by-document matrix $A_{m\times n}$ has the singular value decomposition $A = U\Sigma\ V^T = \sum_{i=1}^{r}\sigma_i\mathbf{u}_i\mathbf{v}_i^T$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, then A_k is created by truncating this expansion after k terms, where k is a user tunable parameter.

2. The recall and precision measures are generally used in conjunction with each other to evaluate performance.

3. A is replaced by $A_k = \sum_{i=1}^{k}\sigma_i\mathbf{u}_i\mathbf{v}_i^T$ in the query process so that if $\mathbf{q}$ and the columns of A_k have been normalized, then the angle vector is computed as $\boldsymbol{\delta}^T = \mathbf{q}^T A_k$.

4. The truncated SVD approximation to A is optimal in the sense that of all rank-k matrices, the truncated SVD A_k is the closest to A, and

$$\|A - A_k\|_F = \min_{rank(B) \leq k} \|A - B\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}.$$

5. This rank-k approximation reduces the so-called linguistic noise present in the term-by-document matrix and thus improves information retrieval performance.

6. [Dum91, BB05, BR99, Ber01, BDJ99] LSI is known to outperform the vector space model in terms of precision and recall.

7. [BR99, Ber01, BB05, BF96, BDJ99, BO98, Blo99, BR01, Dum91, HB00, JL00, JB00, LB97, WB98, ZBR01, ZMS98] LSI and the truncated singular value decomposition dominated text mining research in the 1990s.

8. A serious drawback to LSI is that while it might appear at first glance that A_k should save storage over the original matrix A, this is often not the case, even when $k << r$. This is because A is generally very sparse, but the singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i^T$ are almost always completely dense. In many cases A_k requires more (sometimes much more) storage than A itself requires.

9. A significant problem with LSI is the fact that while A is a nonnegative matrix, the singular vectors are mixed in sign. This loss of important structure means that the truncated singular value decomposition provides no textual or semantic interpretation interpretability. Consider a particular document vector, say, column 1 of A. The truncated singular value decomposition represents document 1 as

$$A_1 = \begin{bmatrix} \vdots \\ \mathbf{u}_1 \\ \vdots \end{bmatrix} \sigma_1 v_{11} + \begin{bmatrix} \vdots \\ \mathbf{u}_2 \\ \vdots \end{bmatrix} \sigma_2 v_{12} + \cdots \begin{bmatrix} \vdots \\ \mathbf{u}_k \\ \vdots \end{bmatrix} \sigma_k v_{1k},$$

so document 1 is a linear combination of the basis vectors $\mathbf{u}_i$ with the scalar $\sigma_i v_{1i}$ being a weight that represents the contribution of basis vector i in document 1. What we'd really like to do is say that basis vector i is mostly concerned with some subset of the terms, but any such textual or semantic interpretation is difficult (or impossible) when SVD components are involved. Moreover, if there were textual or semantic interpretations, the orthogonality of

the singular vectors would ensure that there is no overlap of terms in the topics in the basis vectors, which is highly unrealistic.

10. [Ber01, ZMS98] It is usually a difficult problem to determine the most appropriate value of k for a given dataset because k must be large enough so that A_k can capture the essence of the document collection but small enough to address storage and computational issues. Various heuristics have been developed to deal with this issue.

Examples

1. Consider again the 9×7 term-by-document matrix used in §1. The rank-4 approximation to this matrix is

$$A_4 = \begin{bmatrix} 0.020 & 1.048 & -0.034 & 0.996 & 0.975 & 0.027 & 0.975 \\ -0.154 & 0.883 & 1.067 & 0.078 & 0.027 & -0.033 & 0.027 \\ -0.012 & -0.019 & 0.013 & 0.004 & 0.509 & 0.990 & 0.509 \\ 0.395 & 0.058 & 0.020 & 0.756 & 0.091 & -0.087 & 0.091 \\ -0.154 & 0.883 & 1.067 & 0.078 & 0.027 & -0.033 & 0.027 \\ 0.723 & -0.144 & 0.068 & 1.152 & 0.004 & -0.012 & 0.004 \\ -0.012 & -0.019 & 0.013 & 0.004 & 0.509 & 0.990 & 0.509 \\ 0.443 & 0.334 & 0.810 & 0.776 & -0.074 & 0.091 & -0.074 \\ 0.723 & -0.144 & 0.068 & 1.152 & 0.004 & -0.012 & 0.004 \end{bmatrix}.$$

   Notice that while A is sparse and nonnegative, A_4 is dense and mixed in sign. Of course, as k increases, A_k looks more and more like A.

   For a query on *baby health*, the angle vector is

$$\boldsymbol{\delta} \approx [\, .244 \quad .466 \quad -.006 \quad .564 \quad .619 \quad -.030 \quad .619 \,]^T.$$

   Thus, the information retrieval system returns documents $\mathbf{d}_5$, $\mathbf{d}_7$, $\mathbf{d}_4$, $\mathbf{d}_2$, $\mathbf{d}_1$, in order from most to least relevant. As a result, the recall improves to 2/3, while the precision is 2/5. Adding another singular triplet and using the approximation matrix A_5 does not change the recall or precision measures, but does give a slightly different angle vector

$$\boldsymbol{\delta} \approx [\, .244 \quad .466 \quad -.006 \quad .564 \quad .535 \quad -.030 \quad .535 \,]^T,$$

   which is better than the A_4 angle vector because the most relevant document, $\mathbf{d}_4$, *Your Baby's Health and Safety: From Infant to Toddler*, gets the highest score.

3 Nonnegative Matrix Factorizations

The lack of semantic interpretation due to the mixed signs in the singular vectors is a major obstacle in using LSI. To circumvent this problem, alternative low rank approximations that maintained the

nonnegative structure of the original term-by-document matrix have been proposed [LS99, LS00, PT94, PT97].

Facts:

1. If $A_{m\times n} \geq 0$ has rank r, then for a given $k < r$ the goal of a nonnegative matrix factorization (NMF) is to find the nearest rank-k approximation WH to A such that $W_{m\times k} \geq 0$ and $H_{k\times n} \geq 0$. Once determined, a NMF simply replaces the truncated singular value decomposition in any text mining task such as clustering documents, classifying documents, or processing queries on documents.

2. A NMF can be formulated as a constrained nonlinear least squares problem by first specifying k and then determining

$$\min \|A - WH\|_F^2 \quad \text{subject to} \quad W_{m\times k} \geq 0, \quad H_{k\times n} \geq 0.$$

   The rank of the approximation (i.e., k) becomes the number of topics or clusters in a text mining application.

3. [LS99] The Lee and Seung algorithm to compute a NMF using Matlab is as follows.

Algorithm 1: Lee-Seung NMF

$W = \text{abs}(\text{randn}(m, k))$ % initialize with random dense matrix
$H = \text{abs}(\text{randn}(k, n))$ % initialize with random dense matrix
for $i = 1 :$ maxiter
$H = H. * (W^T A)./(W^T WH + 10^{-9})$ % 10^{-9} avoids division by 0
$W = W. * (AH^T)./(WHH^T + 10^{-9})$
end

4. The objective function $\|A - WH\|_F^2$ in the Lee and Seung algorithm tends to tail off within 50-100 iterations. Faster algorithms exist, but the Lee and Seung algorithm algorithm is guaranteed to converge to a local minimum in a finite number of steps.

5. [Hoy02, Hoy04, SBP04] Other NMF algorithms contain a tunable sparsity parameter that produces any desired level of sparseness in W and H. The storage savings of the NMF over the truncated SVD are substantial.

6. Because $A_j \approx \sum_{i=1}^{k} W_i h_{ij}$, and because W and H are nonnegative, each column W_i can be viewed as a topic vector—if $w_{ij_1}, w_{ij_2}, \ldots, w_{ij_p}$ are the largest entries in W_i, then terms $j_1, j_2, \ldots, j_p$ dictate the topics that W_i represents. The entries h_{ij} measure the strength to which topic i appears in basis document j, and k is the number of topic vectors that one expect to see in a given set of documents.

7. The NMF has some disadvantages. Unlike the SVD, uniqueness and robust computations are missing in the NMF. There is no unique global minimum for the NMF (the defining constrained least squares problem is not convex in W and H), so algorithms can only guarantee convergence to a local minimum, and many don't even guarantee that.

8. Not only will different NMF algorithms produce different NMF factors, the same NMF algorithm, run with slightly different parameters, can produce very different NMF factors. For example, the results can be highly dependent on the initial values.

Examples

1. When the term-by-document matrix of the MEDLINE dataset [Med03] is approximated with a NMF as described above with $k = 10$, the charts in Figure 1 show the highest weighted terms from four representative columns of W. For example, this makes it clear that W_1 represents heart related topics, while W_2 concerns blood issues, etc.

   When document 5 (column A_5) from MEDLINE is expressed as an approximate linear combination of $W_1, W_2, \ldots, W_{10}$ in order of the size of the entries of H_5, which are

$$h_{95} = .1646 > h_{65} = .0103 > h_{75} = .0045 > \cdots,$$

   we have that

$$\begin{aligned} A_5 &\approx .1646\, W_9 + .0103\, W_6 + .0045\, W_7 + \cdots \\ &= .1646 \begin{bmatrix} \text{fatty} \\ \text{glucose} \\ \text{acids} \\ \text{ffa} \\ \text{insulin} \\ \vdots \end{bmatrix} + .0103 \begin{bmatrix} \text{kidney} \\ \text{marrow} \\ \text{dna} \\ \text{cells} \\ \text{nephr.} \\ \vdots \end{bmatrix} + .0045 \begin{bmatrix} \text{hormone} \\ \text{growth} \\ \text{hgh} \\ \text{pituitary} \\ \text{mg} \\ \vdots \end{bmatrix} + \cdots. \end{aligned}$$

   Therefore, document 5 is largely about terms contained in topic vector W_9 followed by topic vectors W_6 and W_7.

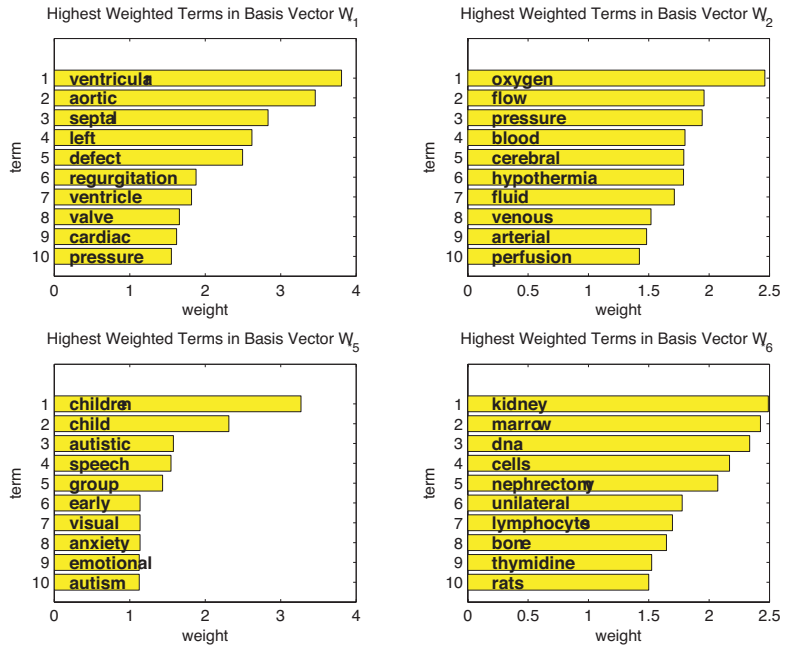

Figure 1: MEDLINE charts

2. Consider the same 9×7 term-by-document matrix A from the example in §1. A rank-4 approximation $A_4 = W_{9\times4}H_{4\times7}$ that is produced by the Lee and Seung algorithm is

$$A_4 = \begin{bmatrix} 0.027 & 0.888 & 0.196 & 1.081 & 0.881 & 0.233 & 0.881 \\ 0 & 0.852 & 1.017 & 0.173 & 0.058 & 0.031 & 0.058 \\ 0.050 & 0.084 & 0.054 & 0.102 & 0.496 & 0.899 & 0.496 \\ 0.360 & 0.172 & 0.073 & 0.729 & 0.179 & 0.029 & 0.179 \\ 0 & 0.852 & 1.017 & 0.173 & 0.058 & 0.031 & 0.058 \\ 0.760 & 0.032 & 0.155 & 1.074 & 0.033 & 0.061 & 0.033 \\ 0.050 & 0.084 & 0.054 & 0.102 & 0.496 & 0.899 & 0.496 \\ 0.445 & 0.481 & 0.647 & 0.718 & 0.047 & 0.053 & 0.047 \\ 0.760 & 0.032 & 0.155 & 1.074 & 0.033 & 0.061 & 0.033 \end{bmatrix},$$

where

$$W_4 = \begin{bmatrix} 0.202 & 0.017 & 0.160 & 1.357 \\ 0 & 0 & 0.907 & 0.104 \\ 0.805 & 0 & 0 & 0.008 \\ 0 & 0.415 & 0 & 0.321 \\ 0 & 0 & 0.907 & 0.104 \\ 0 & 0.875 & 0 & 0.060 \\ 0.805 & 0 & 0 & 0.008 \\ 0 & 0.513 & 0.500 & 0.085 \\ 0 & 0.876 & 0 & 0.060 \end{bmatrix},$$

$$H_4 = \begin{bmatrix} 0.062 & 0.010 & 0.067 & 0.119 & 0.610 & 1.117 & 0.610 \\ 0.868 & 0 & 0.177 & 1.175 & 0 & 0.070 & 0 \\ 0 & 0.878 & 1.121 & 0.105 & 0 & 0.034 & 0 \\ 0 & 0.537 & 0 & 0.752 & 0.559 & 0 & 0.559 \end{bmatrix}.$$

Notice that both A and A_4 are nonnegative, and the sparsity of W and H makes the storage savings apparent. The error in this NMF approximation as measured by $\|A - WH\|_F^2$ is 1.56, while the error in the best rank-4 approximation from the truncated SVD is 1.42. In other words, the NMF approximation is not far from the optimal SVD approximation—this is frequently the case in practice in spite of the fact that W and H can vary with the initial conditions. For a query on *baby health*, the angle vector is

$$\boldsymbol{\delta} \approx [\, .224 \quad .472 \quad .118 \quad .597 \quad .655 \quad .143 \quad .655 \,]^T .$$

Thus, the information retrieval system that uses the nonnegative matrix factorization gives the same ranking as a system that uses the truncated singular value decomposition. However, the factors are sparse and nonnegative, and can be interpreted.

4 Web Search

Only a few years ago users of Web search engines were accustomed to waiting, for what would now seem to be an eternity, for search engines to return results to their queries. And when a search engine finally responded, the returned list was littered with links to information that was either irrelevant, unimportant, or downright useless. Frustration was compounded by the fact that useless links invariably appeared at or near the top of the list while useful links were deeply buried. Users had to sift through links a long way down in the list to have a chance of finding something satisfying, and being less than satisfied was not uncommon.

The reason for this is that the Web's information is not structured like information in the organized databases and document collections that generations of computer scientists had honed their techniques on. The Web is unique in the sense that it is *self organized* . That is, unlike traditional document collections that are accumulated, edited, and categorized by trained specialists, the Web has no standards, no reviewers, and no gatekeepers to police content, structure, and format. Information on the Web is volatile and heterogeneous—links and data are rapidly created, changed, and removed, and Web information exists in multiple formats, languages, and alphabets. And there is a multitude of different purposes for Web data—e.g., some Web pages are designed to inform while others try to sell, cheat, steal, or seduce. In addition, the Web's self organization opens the door for spammers, the nefarious people who want to illicitly commandeer your attention to sell or advertise something to you that you probably don't want. Web spammers are continually devising

diabolical schemes to trick search engines into listing their (or their client's) Web pages near the top of the list of search results. They had an easy time of it when Web search relied on traditional IR methods based on semantic principles. Spammers could create Web pages that contained things such as miniscule or hidden text fonts, hidden text with white fonts on a white background, and misleading metatag descriptions designed to influence semantic based search engines. Finally, the enormous size of the Web, currently containing $O(10^9)$ pages, completely overwhelmed traditional IR techniques.

By 1997 it was clear that in nearly all respects the database and IR technology of the past wasn't well suited for Web search, so researchers set out to devise new approaches. Two big ideas emerged (almost simultaneously), and each capitalizes on the link structure of the Web to differentiate between relevant information and fluff. One approach, HITS (Hypertext Induced Topic Search), was introduced by Jon Kleinberg [Kle99, LM06], and the other, which changed everything, is Google's *PageRank* that was developed by Sergey Brin and Larry Page [BP98, BPM99, LM06]. While variations of HITS and PageRank followed (e.g., Lempel's SALSA [LM00, LM05, LM06]), the basic idea of PageRank became the driving force, so the focus is on this concept.

Definitions:

Early in the game search companies such as YAHOO! employed students to surf the Web and record key information about the pages they visited. This quickly overwhelmed human capability, so today all Web search engines use **Web Crawlers,** which is software that continuously scours the Web for information to return to a central repository.

Web pages found by the robots are temporarily stored in entirety in a **page repository.** Pages remain in the repository until they are sent to an indexing module, where their vital information is stripped to create a compressed version of the page. Popular pages that are repeatedly requested by users are stored here longer, perhaps indefinitely.

The **indexing module** extracts only key words, key phrases, or other vital descriptors, and it creates a compressed description of the page that can be "indexed." Depending on the popularity of a page the uncompressed version is either deleted or returned to the page repository.

There are three general kinds of **indexes** that contain compressed information for each webpage. The **content index** contains information such as key words or phrases, titles, and anchor text, and this is stored in a compressed form using an **inverted file** structure, which is simply the electronic version of a book index—i.e., each morsel of information points to a list of pages containing

it. Information regarding the hyperlink structure of a page is stored in compressed form in the **structure index**. The crawler module sometimes accesses the structure index to find uncrawled pages. Finally, there are **special-purpose indexes** such as an image index and a pdf index. The crawler, page repository, indexing module, and indexes, along with their corresponding data files exist and operate independent of users and their queries.

The **query module** converts a user's natural language query into a language that the search engine can understand (usually numbers), and consults the various indexes in order to answer the query. For example, the query module consults the content index and its inverted file to find which pages contain the query terms.

The **pertinent pages** are the pages that contain query terms. After pertinent pages have been identified, the query module passes control to the ranking module.

The **ranking module** takes the set of pertinent pages and ranks them according to some criterion, and this criterion is the heart of the search engine—it is the distinguishing characteristic that differentiates one search engine from another. The ranking criterion must somehow discern which Web pages best respond to a user's query, a daunting task because there might be millions of pertinent pages. Unless a search engine wants to play the part of a censor (which most don't), the user is given the opportunity of seeing a list of links to a large proportion of the pertinent pages, but with less useful links permuted downward.

PageRank is Google's patented ranking system, and some of the details surrounding PageRank are discussed below.

Facts:

1. Google assigns at least two scores to each Web page. The first is a popularity score and the second is a content score. Google blends these two scores to determine the final ranking of the results that are returned in response to a user's query.

2. [BP98] The rules used to give each pertinent page a content score are trade secrets, but they generally take into account things such as whether the query terms appear in the title or deep in the body of a Web page, the number of times query terms appear in a page, the proximity of multiple query words to one another, and the appearance of query terms in a page (e.g., headings in bold font score higher). The content of neighboring web pages is also taken into account.

3. Google is known to employ over a hundred such metrics in this regard, but the details are proprietary. While these metrics are important, they are secondary to the the popularity score, which is the primary component of PageRank. The content score is used by Google only to temper the popularity score.

5 Google's PageRank

The popularity score is where the mathematics lies, so it is the focus of the remainder of this exposition. We will identify the term "PageRank" with just the mathematical component of Google's PageRank (the popularity score) with the understanding that PageRank may be tweaked by a content score to produce a final ranking.

Both PageRank and Google were conceived by Sergey Brin and Larry Page while they were computer science graduate students at Stanford University, and in 1998 they took a leave of absence to focus on their growing business. In a public presentation at the Seventh International World Wide Web conference (WWW98) in Brisbane, Australia, their paper "The PageRank citation ranking: Bringing order to the Web" [BPM99] made small ripples in the information science community that quickly turned into waves.

The original idea was that a *page is important if it is pointed to by other important pages.* That is, the importance of your page (its PageRank) is determined by summing the PageRanks of all pages that point to yours. Brin and Page also reasoned that when an important page points to several places, its weight (PageRank) should be distributed proportionately.

In other words, if YAHOO! points to 99 pages in addition to yours, then you should only get credit for 1/100 of YAHOO!'s PageRank. This is the intuitive motivation behind Google's PageRank concept, but significant modifications are required to turn this basic idea into something that works in practice.

For readers who want to know more, the book *Google's PageRank and Beyond: The Science of Search Engine Rankings* [LM06] from Princeton University Press contains over 250 pages devoted to link analysis algorithms along with other ranking schemes such as HITS and SALSA as well as additional background material, examples, code, and chapters dealing with more advanced issues in Web search ranking.

Definitions:

The **hyperlink matrix** is the matrix $H_{n\times n}$ that represents the link structure of the Web, and its entries are given by

$$h_{ij} = \begin{cases} 1/|O_i| & \text{if there is a link from page } i \text{ to page } j, \\ 0 & \text{otherwise,} \end{cases}$$

where $|O_j|$ is the number of outlinks from page j.

Suppose that there are n Web pages, and let $r_i(0)$ denote the initial rank of the i^{th} page. If the ranks are successively modified by setting

$$r_i(k+1) = \sum_{j\in I_i} \frac{r_j(k)}{|O_j|}, \quad k = 1, 2, 3, \ldots,$$

where $r_i(k)$ is the rank of page i at iteration k and I_i is the set of pages pointing (linking) to page i, then the rankings after the k^{th} iteration are

$$\mathbf{r}^T(k) = \Big(r_1(k),\, r_2(k),\, \ldots,\, r_n(k)\Big) = \mathbf{r}^T(0)H^k.$$

The **conceptual PageRank** of the i^{th} Web page is defined to be

$$r_i = \lim_{k\to\infty} r_i(k),$$

provided that the limit exists. However, this definition is strictly an intuitive concept because the natural structure of the Web generally prohibits these limits from existing.

A **dangling node** is a Web page that contain no out links. Dangling nodes produce zero rows in the hyperlink matrix H, so even if $\lim_{k\to\infty} H^k$ exists, the limiting vector $\mathbf{r}^T = \lim_{k\to\infty} \mathbf{r}^T(k)$ would be dependent on the initial vector $\mathbf{r}^T(0)$, which is not good.

The **stochastic hyperlink matrix** is produced by perturbing the hyperlink matrix to be stochastic. In particular,

$$S = H + \mathbf{a}\mathbf{1}^T/n, \tag{1}$$

where $\mathbf{a}$ is the column in which

$$a_i = \begin{cases} 1 & \text{if page } i \text{ is a dangling node,} \\ 0 & \text{otherwise.} \end{cases}$$

S is a stochastic matrix that is identical to H except that zero rows in H are replaced by $\mathbf{1}^T/n$ ($\mathbf{1}$ is a vector of 1s and $n = O(10^9)$, so entries in $\mathbf{1}^T/n$ are pretty small). The effect is to eliminate dangling nodes. Any probability vector $\mathbf{p}^T > 0$ can be used in place of the uniform vector.

The **Google matrix** is defined to be the stochastic matrix

$$G = \alpha S + (1 - \alpha)E, \tag{2}$$

where $E = \mathbf{1}\mathbf{v}^T$ in which $\mathbf{v}^T > 0$ can be any probability vector. Google originally set $\alpha = .85$ and $\mathbf{v}^T = \mathbf{1}^T/n$. The choice of α is discussed later in facts 12, 13, and 14.

The **personalization vector** is the vector $\mathbf{v}^T$ in $E = \mathbf{1}\mathbf{v}^T$ in (2). Manipulating $\mathbf{v}^T$ gives Google the flexibility to make adjustments to PageRanks as well as to personalize them (thus the name "personalization vector") [HKJ03, LM04a].

The **PageRank vector** is the left Perron vector $\boldsymbol{\pi}^T$ of the Google matrix G. In particular, $\boldsymbol{\pi}^T(I - G) = 0$, where $\boldsymbol{\pi}^T > 0$ and $\|\boldsymbol{\pi}^T\|_1 = 1$. The components of this vector constitute Google's popularity score of each webpage.

Facts:

1. [Mey00, Chapt. 8] The Google matrix G is a primitive stochastic matrix, so the spectral radius $\rho(G) = 1$ is a simple eigenvalue, and 1 is the only eigenvalue on the unit circle.

2. The iteration defined by $\boldsymbol{\pi}^T(k+1) = \boldsymbol{\pi}^T(k)G$ converges, independent of the starting vector, to a unique stationary probability distribution $\boldsymbol{\pi}^T$, which is the PageRank vector.

3. The irreducible aperiodic Markov chain defined by $\boldsymbol{\pi}^T(k+1) = \boldsymbol{\pi}^T(k)G$ is a constrained random walk on the Web graph. The random walker can be characterized as a Web surfer who, at each step, randomly chooses a link from his current page to click on except that:

   (a) When a dangling node is encountered, the excursion is continued by jumping to another page selected at random (i.e., with probability $1/n$).

   (b) At each step the random Web surfer has a chance (with probability $1 - \alpha$) of becoming bored with following links from the current page, in which case the random Web surfer continues the excursion by jumping to page j with probability v_j.

4. The random walk defined by $\mathbf{r}^T(k+1) = \mathbf{r}^T(k)S$ will generally not converge because Web's graph structure is not strongly connected, which results in a reducible chain with many isolated ergodic classes.

5. The power method has been Google's computational method of choice for computing the PageRank vector. If formed explicitly, G is completely dense, and the size of n would make

each power iteration extremely costly—billions of flops for each step. But writing the power method as

$$\boldsymbol{\pi}^T(k+1) = \boldsymbol{\pi}^T(k)G = \alpha\boldsymbol{\pi}^T(k)H + \Big(\alpha\boldsymbol{\pi}^T(k)\mathbf{a}\Big)\mathbf{1}^T/n + (1-\alpha)\mathbf{v}^T$$

shows that only extremely sparse vector-matrix multiplications are required. Only the nonzero h_{ij}'s are needed, so G and S are neither formed nor stored.

6. When implemented as shown above, each power step requires only $nnz(H)$ flops, where $nnz(H)$ is the number of nonzeros in H, and, since the average number of nonzeros per column in H is significantly less than 10, we have $O(nnz(H)) \approx O(n)$. Furthermore, the inherent parallism is easily exploited, and the current iterate is the only vector stored at each step.

7. Because the Web has many disconnected components, the hyperlink matrix is highly reducible, and compensating for the dangling nodes to construct the stochastic matrix S does not significantly affect this.

8. [Mey00, p. 695-696] Since S is also reducible, S can be symmetrically permuted to have the form

$$S \sim \left[\begin{array}{cccc|cccc} S_{11} & S_{12} & \cdots & S_{1r} & S_{1,r+1} & S_{1,r+2} & \cdots & S_{1m} \\ 0 & S_{22} & \cdots & S_{2r} & S_{2,r+1} & S_{2,r+2} & \cdots & S_{2m} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & S_{rr} & S_{r,r+1} & S_{r,r+2} & \cdots & S_{rm} \\ \hline 0 & 0 & \cdots & 0 & S_{r+1,r+1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & S_{r+2,r+2} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & S_{mm} \end{array}\right],$$

where the following are true.

- For each $1 \le i \le r$, S_{ii} is either irreducible or $[0]_{1\times 1}$.
- For each $1 \le i \le r$, there exists some $j > i$ such that $S_{ij} \neq 0$.
- $\rho(S_{ii}) < 1$ for each $1 \le i \le r$.
- $S_{r+1,r+1}$, $S_{r+2,r+2}, \cdots, S_{m,m}$ are each stochastic and irreducible.
- 1 is an eigenvalue for S that is repeated exactly $m - r$ times.

9. The natural structure of the Web forces the algebraic multiplicity of the eigenvalue 1 to be large.

10. [LM04a, LM06, Mey00, Ex. 7.1.17, p. 502] If the eigenvalues of $S_{n\times n}$ are

$$\lambda(S) = \{\underbrace{1, 1, \ldots, 1}_{m-r}, \mu_{m-r+1}, \ldots, \mu_n\}, \quad 1 > |\mu_{m-r+1}| \geq \cdots \geq |\mu_n|,$$

then the eigenvalues of the Google matrix $G = \alpha S + (1-\alpha)E$ are

$$\lambda(G) = \{1, \underbrace{\alpha, \ldots, \alpha}_{m-r-1}, (\alpha\mu_{m-r+1}), \ldots, (\alpha\mu_n)\}. \tag{3}$$

11. [Mey00] The asymptotic rate of convergence of any aperiodic Markov chain is governed by the magnitude of its largest subdominant eigenvalue. In particular, if the distinct eigenvalues λ_i of an aperiodic chain are ordered $\lambda_1 = 1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$, then the number of incorrect digits in each component of $\boldsymbol{\pi}^T(k)$ is eventually going to be reduced by about $-log_{10}|\lambda_2|$ digits per iteration.

12. Determining (or even estimating) $|\lambda_2|$ normally requires substantial effort, but equation (3) says that $\lambda_2 = \alpha$ for the Google matrix G. This is an extremely happy accident because it means that Google's engineers can completely control the rate of convergence, regardless of the value of the personalization vector $\mathbf{v}^T$ in $E = \mathbf{1}\mathbf{v}^T$.

13. At the last public disclosure Google was setting $\alpha = .85$, so, at this value, the asymptotic rate of convergence of the power method is $-log_{10}(.85) \approx .07$, which means that the power method will eventually take about 14 iterations for each significant place of accuracy that is required.

14. [LM04a] Even though they can control the rate of convergence with α, Google's engineers are forced to perform a delicate balancing act because while decreasing α increases the rate of convergence, decreasing α also lessens the effect of the true hyperlink structure of the Web, which is Google's primary mechanism for measuring webpage importance. Increasing α more accurately reflects the Web's true link structure, but, along with slower convergence, sensitivity issues begin to surface in the sense that slightly different values for α near 1 can produce significantly different PageRanks.

15. [KHM03a] The power method can be substantially accelerated by a quadratic extrapolation technique similar to Aitken's Δ^2 method, and there is reason to believe that Google has adopted this procedure.

16. [KHM03b, KHG04] Other improvements to the basic power method include a block algorithm and an adaptive algorithm that monitors the convergence of individual elements of the PageRank vector. As soon as components of the vector have converged to an acceptable tolerance, they are no longer computed. Convergence is faster because the work per iteration decreases as the process evolves.

17. [LM02, LM04a, LM04b, LGZ03] Other methods partition H into two groups according to dangling nodes and nondangling nodes. The problem is then aggregated by lumping all of the dangling nodes into one super state to produce a problem that is significantly reduced in size—this is due to the fact that the dangling nodes account for about one-fourth of the Web's nodes. The most exciting feature of all these algorithms is that they do not compete with one another. In fact, it is possible to combine some of these algorithms to achieve even greater performance.

18. The accuracy of PageRank computations is an important implementation issue, but we do not know the accuracy with which Google works. It seems that it must be at least high enough to differentiate between the often large list of ranked pages that Google usually returns, and since $\boldsymbol{\pi}^T$ is a probability vector containing $O(10^9)$ components, it is reasonable to expect that accuracy giving at least ten significant places is needed to distinguish among the elements.

19. A weakness of PageRank is "topic drift." The PageRank vector might be mathematically accurate, but this is of little consequence if the results point the user to sites that are off-topic. PageRank is a query-independent measure that is essentially determined by a popularity contest with everyone on Web having a vote, and this tends to skew the results in the direction of importance (measured by popularity) over relevance to the query. This means that PageRank may have trouble distinguishing between pages that are authoritative in general and pages that are authoritative more specifically to the query topic. It is believed that Google engineers devote much effort to mitigate this problem, and this is where the metrics that determine the content score might have an effect.

20. In spite of topic drift, Google's decision to measure importance by means of popularity over relevance turned out to be the key to Google's success and the source of its strength. The query dependent measures employed by Google's predecessors were major stumbling blocks in maintaining query processing speed as the Web grew. PageRank is query-independent,

so at query time only a quick lookup into an inverted file storage is required to determine pertinent pages, which are then returned in order the precomputed PageRanks. The small compromise of topic drift in favor of processing speed won the day.

21. A huge advantage of PageRank is its virtual imperviousness to spamming (artificially gaming the system). High PageRank is achieved by having many inlinks from highly ranked pages, and while it may not be so hard for a page owner to have many of his cronies link to his page, it is difficult to generate a lot inlinks from *important* sites that have a high rank.

22. [TM03] Another strength of PageRank concerns the flexibility of the "personalization" (or "intervention") vector $\mathbf{v}^T$ that Google is free to choose when defining the perturbation term $E = \mathbf{1}\mathbf{v}^T$. The choice of $\mathbf{v}^T$ affects neither mathematical nor computational aspects, but it does alter the rankings in a predictable manner. This can be a terrific advantage if Google wants to intervene to push a site's PageRank down or up, perhaps to punish a suspected "link farmer" or to reward a favored client. Google has claimed that they do not make a practice of rewarding favored clients, but it is known that Google is extremely vigilant and sensitive concerning people who try to manipulate PageRank, and such sites are punished. However, the outside world is not privy to the extent to which either the stick or carrot is employed.

23. [FLM+04, KH03, KHG04, KHM03a, LM04a, LM06, LGZ03, LM03, NZJ01] In spite of the simplicity of the basic concepts, subtle issues such as personalization, practical implementations, sensitivity, and updating lurk just below the surface.

24. We have summarized only the mathematical component of Google's ranking system, but, as mentioned earlier, there are a hundred or more non-mathematical content metrics that are also considered when Google responds to a query. The results seen by a user are in fact PageRank tempered by these other metrics. While Google is secretive about most of their other metrics, they have made it clear that the other metrics are subservient to their mathematical PageRank scores.

References

[BB05] Michael W. Berry and Murray Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. SIAM, Philadelphia, 2nd edition, 2005.

[BDJ99] Michael W. Berry, Zlatko Drmac, and Elizabeth R. Jessup. Matrices, vector spaces and information retrieval. *SIAM Review*, 41:335–362, 1999.

[Ber01] Michael W. Berry, editor. *Computational Information Retrieval*. SIAM, Philadelphia, 2001.

[BF96] Michael W. Berry and R. D. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Journal of Numerical Linear Algebra with Applications*, 1(1):1–27, 1996.

[Blo99] Katarina Blom. *Information retrieval using the singular value decomposition and Krylov subspaces*. PhD thesis, University of Chalmers, January 1999.

[BO98] Michael W. Berry and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1998.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 33:107–117, 1998.

[BPM99] Sergey Brin, Lawrence Page, R. Motwami, and Terry Winograd. The PageRank citation ranking: bringing order to the Web. Technical Report 1999-0120, Computer Science Department, Stanford University, 1999.

[BR01] Katarina Blom and Axel Ruhe. Information retrieval using very short Krylov sequences. In *Computational Information Retrieval*, pages 41–56, 2001.

[BR99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.

[Dum91] Susan T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23:229–236, 1991.

[FLM+04] Ayman Farahat, Thomas Lofaro, Joel C. Miller, Gregory Rae, and Lesley A. Ward. Existence and uniqueness of ranking vectors for linear link analysis. *SIAM Journal on Scientific Computing*, 2004. submitted April 2004.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.

[HB00] M. K. Hughey and Michael W. Berry. Improved query matching using kd-trees, a latent semantic indexing enhancement. *Information Retrieval*, 2:287–302, 2000.

[HKJ03] Taher H. Haveliwala, Sepandar D. Kamvar, and Glen Jeh. An analytical comparison of approaches to personalizing PageRank. Technical report, Stanford University, 2003.

[Hoy02] Patrik O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII (Proc. IEEE Workshop on Neural Networks for Signal Processing)*, pages 557–565, 2002.

[Hoy04] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[JB00] Eric P. Jiang and Michael W. Berry. Solving total least squares problems in information retrieval. *Linear Algebra and its Applications*, 316:137–156, 2000.

[JL00] Fan Jiang and Michael L. Littman. Approximate dimension equalization in vector-based information retrieval. In *The Seventeenth International Conference on Machine Learning*, pages 423–430, 2000.

[KH03] Sepandar D. Kamvar and Taher H. Haveliwala. The condition number of the PageRank problem. Technical report, Stanford University, 2003.

[KHG04] Sepandar D. Kamvar, Taher H. Haveliwala, and Gene H. Golub. Adaptive methods for the computation of PageRank. *Linear Algebra and its Applications*, 386:51–65, 2004.

[KHM03a] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating PageRank computations. In *Twelfth International World Wide Web Conference*, New York, NY, 2003. ACM Press.

[KHM03b] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Exploiting the block structure of the Web for computing PageRank. Technical Report 2003-17, Stanford University, 2003.

[Kle99] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46, 1999.

[LB97] Todd A. Letsche and Michael W. Berry. Large-scale information retrieval with LSI. *Informatics and Computer Science*, pages 105–137, 1997.

[LGZ03] Chris Pan-Chi Lee, Gene H. Golub, and Stefanos A. Zenios. A fast two-stage algorithm for computing PageRank and its extensions. Technical Report SCCM-2003-15, Scientific Computation and Computational Mathematics, Stanford University, 2003.

[LM00] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *The Ninth International World Wide Web Conference*, New York, NY, 2000. ACM Press.

[LM02] Amy N. Langville and Carl D. Meyer. Updating the stationary vector of an irreducible Markov chain. Technical Report crsc02-tr33, N. C. State, Mathematics Dept., CRSC, 2002.

[LM03] Ronny Lempel and Shlomo Moran. Rank-stability and rank-similarity of link-based web ranking algorithms in authority-connected graphs. In *Second Workshop on Algorithms and Models for the Web-Graph (WAW 2003)*, Budapest, Hungary, May 2003.

[LM04a] Amy N. Langville and Carl D. Meyer. Deeper inside PageRank. *Internet Mathematics Journal*, 2004. Accepted in February 2004.

[LM04b] Amy N. Langville and Carl D. Meyer. A reordering for the PageRank problem. *SIAM Journal on Scientific Computing*, 2004. submitted August.

[LM05] Amy N. Langville and Carl D. Meyer. A survey of eigenvector methods of web information retrieval. *The SIAM Review*, 47(1):135–161, 2005.

[LM06] Amy N. Langville and Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ, 2006.

[LS99] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[LS00] Daniel D. Lee and H. Sebastian Seung. Algorithms for the non-negative matrix factorization. *Advances in Neural Information Processing*, 2000.

[Med03] Medlars test collection, 2003. `http://www.cs.utk.edu/~lsi/`.

[Mey00] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2000.

[NZJ01] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Link analysis, eigenvectors and stability. In *The Seventh International Joint Conference on Artificial Intelligence*, 2001.

[PT94] Pentti Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[PT97] Pentti Paatero and U. Tapper. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37:23–35, 1997.

[SB83] Gerard Salton and Chris Buckley. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[SBP04] Farial Shahnaz, Michael W. Berry, V. Paul Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Journal on Information Processing and Management*, 2004. to appear.

[TM03] Michael Totty and Mylene Mangalindan. As Google becomes web's gatekeeper, sites fight to get in. *Wall Street Journal*, CCXLI(39), 2003. February 26.

[WB98] Dian I. Witter and Michael W. Berry. Downdating the latent semantic indexing model for conceptual information retrieval. *The Computer Journal*, 41(1):589–601, 1998.

[ZBR01] Xiaoyan Zhang, Michael W. Berry, and Padma Raghavan. Level search schemes for information filtering and retrieval. *Information Processing and Management*, 37:313–334, 2001.

[ZMS98] Hongyuan Zha, Osni Marques, and Horst D. Simon. A subspace-based model for information retrieval with applications in latent semantic indexing. *Lecture Notes in Computer Science*, 1457:29–42, 1998.