Detecting Similar Documents Using Salient Terms

James W. Cooper, Anni R. Coden, Eric W. Brown

IBM T J Watson Research Center
PO Box 704, Yorktown Heights, NY 10598
{jwcnmr, anni, brown} @ watson.ibm.com

ABSTRACT

We describe a system for rapidly determining document similarity among a set of documents obtained from an information retrieval (IR) system. We obtain a ranked list of the most important terms in each document using a rapid phrase recognizer system. We store these in a database and compute document similarity using a simple database query. If the number of terms found to not be contained in both documents is less than some predetermined threshold compared to the total number of terms in the document, these documents are determined to be very similar. We compare this to the shingles approach.

Categories and Subject Descriptors

H3.3 [**Information Search and Retrieval**] Clustering, Information Filtering, Selection process.

General Terms

Algorithms, Measurement, Documentation, Performance, Design

Keywords

Text mining, Duplicate documents, Databases, Shingles, Document similarity.

1. INTRODUCTION

One of the continuing problems in information retrieval is the fact that in the web environment, there are a large number of near-duplicate documents returned from most searches. A number of methods have been proposed for recognizing and eliminating such duplicates.

For example, Brown and Prager [1] note that documents having similar rank and identical metadata such as size, date, and base filename are likely to be copies kept on different directories or on different servers and can effectively be reduced to one single occurrence.

A more sophisticated system was described by Broder [2], in which regions of each document, called "shingles" are each treated as a sequence of tokens and then reduced to a numerical representation. These are then converted to "fingerprints" using a method originally described by Rabin [3]. Since the number of identical tokens could be compared, a document similarity measure could also be computed.

At a more simplistic level, Bloomfield [4] has recently described an algorithm for detecting plagiarism in which he simply searches for matches of six or more successive words between two documents. This provides as measure of identical embedded document sections, but not necessarily of document equality.

In this paper, we analyze the technique of characterizing documents using the major multi-word terms discovered in them using phrase recognition software, and develop the hypothesis that documents which have identical lists of discovered terms are effectively identical in content. In this context we use "terms" to mean multi-word terms and salient single word terms.

In the first section we discuss the text mining technology we use, and how we manage the data. In the next section we discuss how we define duplicate documents and how we detect them using the results of text mining. We describe the shingles algorithm and throughout compare this term-based technique with the shingles technique. Then in the following section, we propose a novel "document signature" for the rapid comparison of documents, and finally we discuss how this technique can be used quite successfully for finding documents that are variations on the original document.

2. THE TALENT TEXT MINING SYSTEM

In approaching document comparisons, we utilized the suite of text analysis tools collectively known as Talent (Text Analysis and Language Engineering Tools) for analyzing all the documents in the collection.

2.1 Textract Phrase Extraction

The primary tool we use for analyzing documents is **Textract**, itself a chain of tools for recognizing multi-word terms and proper names. We have described the features of Textract previously [5], [6]. Textract recognizes named entities [7], multi-word terms [8] and named [9] and unnamed relations between terms.

Textract reduces related forms of a term to a single *canonical form* that it can then use in computing term occurrence statistics more accurately. In addition, it recognizes abbreviations and finds the canonical forms of the words they stand for and aggregates these terms into a vocabulary for the entire collection, and for each document, keeping both document and collection-level statistics on these terms. Each term is given a collection-level importance ranking called the IQ or Information Quotient [5], [14]. IQ is effectively a measure of the document selectivity of a particular term: a term that appears in only a few documents is highly selective and has a high IQ. On the other hand, a term that appears in many documents is far less selective and has a low IQ. We note that IQ may not be an especially significant metric across the entire web, where many domains are discussed, but within a single domain, or within the results of a specific search, it can be quite useful.

Textract further categorizes the entities it discovers into one of the categories shown in Table 1. The earlier categories have the least certainty and the later ones higher certainty.

Table 1 –Categories assigned to terms by Textract

UWORD	Unknown word
UTERM	Unknown term
UABBR	Unknown abbreviation
UNAME	Unknown type of name
PLACE?	Probably a place
PERSON?	Probably a person
PLACE	A place
PERSON	A person
ORG	An organization

While Textract is our tool of choice in these experiments, we note that there are a number of other systems that have been developed for phrase recognition. For example, Liddy has developed DR-LINK, which is marketed by Textwise [10], and Evans (et al) have developed LinkIT [11]. Further, somewhat similar technology is available in the ThingFinder program from InXight [12].

2.2 Data Management

Once the data from a collection of documents have been analyzed using Textract, it is useful to construct a system for managing that data in a way that makes it easy to sort the terms by document, by IQ and by term type. For this purpose, we constructed a Java class library known as KSS [15], [16] (for Knowledge System Server), which builds a database from the Textract data.

3. DUPLICATE OR SIMILAR DOCUMENTS

In any web search, it is fairly likely that some documents that are returned are very similar. For example, the same documents may exist on several servers or in several users' directories. Some very frequently found examples of this are the Java API documents from Sun, which are found on almost every Java developer's machine. Since these are very well known and described, they are very easy to eliminate using any of the existing techniques.

However, more difficult cases occur when

- There are several edited versions of the same document on various servers.
- The same document is found in several forms, such as HTML and PDF.
- A document is embedded in another. In this case the embedded document may or may not be the most significant part of the combined document.

These cases are more difficult to solve rapidly and it is these that make up the subject of this study.

For the purposes of this study, we define duplicate documents as ones that have essentially the same words in the same sentences and paragraphs. These paragraphs and their sentences could be in a somewhat different order, however. The documents may be in different forms, such as HTML and PDF and they may vary slightly in header boiler-plate information. We define *similar* documents as those in which a large percentage of the sentences, or words in the sentences, are the same.

4. THE SHINGLES TECHNIQUE

We will be contrasting our findings with those obtained using the well-known shingles technique [2]. This technique amounts to reducing each document to a series of numeric codes, such as hash codes, based on sequences of words. In the original paper, the authors suggested making each hash code of a group of 10 adjacent words, and moving the window by one word to create the next hash code. They then eliminate duplicates and, to reduce the number of values, save only those divisible by 25. If this is still too many, they save only the 400 smallest values. The advantage of using shingles to compare documents is that a simple set membership between two tables of integers can be computed very rapidly. Documents that match in all shingles are assumed to be identical and those that match nearly all shingles are closely related.

5. CURRENT EXPERIMENTS

Given the array of sophisticated term management technologies our group has developed, we undertook to find out whether these systems can be use to detect document similarity. In particular, is it possible to use some subset of terms found by Textract as a compressed document representation, which we can use to make rapid comparisons of documents and cluster or eliminate those that are essentially identical?

5.1 Query 1: Detecting Similar Documents

In our first experiment, we went to a popular search engine site with all enhancements turned off, and issued the query "fix broken Thinkpad." This is the sort of naïve query that returns a plethora of documents the user does not want, or expect. Much as we predicted, there were *no* documents on how to repair Thinkpads. However, many of the top 50 documents contained all of these terms in some context or other. Of the top 50, we were able to download and analyze 36 of them. The remainder were broken links. Ten of these documents were Adobe Acrobat PDF files. We used the Gemini plug-in [17] for Adobe Acrobat Reader to export these files into HTML.

We then created a single collection of these documents and analyzed it using Textract. Textract produces a file of terms found in each document, and a file of all the terms in the collection along with their IQ. We used the KSS Java libraries to load these result files into DB2 and subjected the results to various SQL queries to determine the number of terms that documents had in common.

5.2 Similarity Queries

We first must find the significant terms in each document. Initially, we ranked all the terms except the unknown word types in order of decreasing IQ. We note that the IQ measure is a collection level statistic, and that this processing must be done on an entire group of returned documents for this measure to be meaningful.

The question we then want to ask, then, is which terms are not found in common between pairs of documents. You can find these in a single SQL query of the sort

```
Select count(*) as c from
  (select terms from doc1 where ..)
not in
    (select terms from doc2 where ..)
```

After some experimentation, we determined that the important selection criteria are to select terms with an IQ > 0 and which were not UWORDS.

This returns the count of the number of terms that appear in document 2 that are not in document 1. While it might seem that n^2 queries are necessary, it is really only necessary to traverse the upper triangle of this matrix. We do this by sorting the documents into order of increasing size, and comparing documents with the next larger ones in the list. The comparisons are done rapidly in a single database query, and we further reduce the number of compares by limiting the test to documents that are no more than about 10% larger than the compared document. This parameter is, of course, adjustable.

5.3 Results

We found 6 clusters of documents in the 36 documents we analyzed in the first query. Three of these were pairs of identical documents returned from different servers, as shown in Table 2. These documents are identical by any measure and can be easily recognized and collapsed to a single entity, using the methods described by Brown and Prager [1].

Table 3 contains an interesting cluster of eight documents that have similar names, but different sizes. The final two columns of the table shows the fractional difference in contained terms and shingles between adjacent documents in the list.

It is easy to see that these documents must be closely related versions of the same information. In fact, they are all different versions of the same IBM manual describing the Websphere server product. They differ in small details: for example one manual mentions the Linux version of Websphere and another does not. Each of these documents was returned as a PDF file and was converted to HTML using the Gemini plug-in mentioned above.

In Table 3, documents 1 and 2 and documents 3 and 4 are almost certainly absolutely identical. However, the remaining four documents are clearly all closely similar as well. This algorithm finds such documents much as shingles does, but with the added advantage that you can find out which salient terms appear in one and not in the other.

With this term information, one could thus remove some document from the cluster of very similar documents if the terms that are different between them include a term that is also contained in the original query.

Table 2– Identical documents returned by Query 1

Document name	Size	Term diff	Shingle diff
Sytpf130, Sytpfl1301	12236	0	0
aixy2k. aixy2k1	153255	0	0
Conf, client	52854	0	0

Table 3– Very Similar Documents returned from Query 1

#	Title	Size	Terms	Delta terms	Shingl e Delta
1	Fund2	481963	2198	--	--
2	Fund4	481963	2207	0	0
3	ct7mhna1	493225	2139	0.014	0.112
4	ct7mhna2	493295	2146	0	0
5	Fund0	503138	2235	.029	0.074
6	Fund1	504737	2249	.016	0.120
7	Fund3	505004	2287	.011	0.094
8	Fund5	505471	2271	.005	0.012

This search also returned two other closely related document pairs as shown in Table 4. the two documents are in fact a draft and a final PDF document of a paper by Selker and Burleson. [18] Since these papers are quite different in size and format, they would probably not have been found as similar by simple size and filename methods. The term differences between the two are partly because of some additional abstract and polishing, and partly because of the included boilerplate from the magazine format. However, it is surprising to note that the shingles technique does not suggest that these are similar at all, if we assume that a difference of 10% or less is required to call documents similar. We note that the papers each contain about 7800 single words after markup is removed, and that Textract finds 169 multi-words and about 1300 total terms in each of them.

Table 4 – Similar Documents from Query 1

Title	Size	Terms	Term diff	Shingles diff
Selhtm.htm	50169	257	--	--
Selpdf.htm	54274	218	.106	0.44

Careful examination of these two documents and their shingles led to three explanations for the differences:

1. The PDF document contains a table of contents and running page numbers which do not appear in the HTML version.
2. The hyphenation of the PDF document is different and more tokens are created containing trailing hyphens. We changed our shingling program to remove these hyphens and recombine the words, with little improvement.
3. The Gemini PDF translator converts hyphens to "soft hyphens" instead of "hard hyphens," and these have different character codes.

Replacing the hyphen character codes (0xad) with the hard hyphens (0x2d) resulted only in a slight change in the shingles similarity, reducing 0.44 to 0.39. Thus, the documents are genuinely different at the word-by-word level tracked by shingles, but the same using the salient term comparison approach.

6. DETECTING IDENTICAL DOCUMENTS

When documents are very large, it is not usually convenient to run phrase recognition software on the entire set of documents in real time when they are returned from a query, because the elapsed time is too great. However, as part of the indexing process, it is not unreasonable to catalog the major terms in each document. However, even making comparisons among large numbers of terms in multiple documents can take many seconds and can lead to unacceptable delays in returning answers.

We suggest that it is possible to compute a digital signature of each document, based on the terms we find in it. Such a signature can be as simple as a sum of the hash codes of the term strings that make up these major terms. In this experiment, we used the Java String class's hashCode method to compute a code for each term found in a document, and then added these codes up to form the signature. The results are shown in Table 5. We note that this simplification ignores the term order, and could in the pathological extreme possibly result in documents that differed by a negative stop-word such as "not," thus conceivably equating documents which reach opposite conclusions.

Table 5- Computed document signatures

Url	Size	Term Signature	Shingle signature
Fund2	481963	24681058826	**-846170517750**
Fund4	481963	26371644438	**-846170517750**
Ct7mhna1	493225	33130506660	***-852736297100***
Ct7mhna2	493295	32920545738	***-852736297100***
Fund0	503138	10451017932	-861022256825
Fund1	504737	8933271588	**-865769789850**
Fund3	505004	7211814280	**-865769789850**
Fund5	505471	12114748848	-861022256825
Sytpf130	12236	**13802917585**	**1506364450**
Sytpf1301	12236	**13802917585**	**1506364450**
aixy2k	153255	***28232730155***	***-128905766325***
aixy2k1	153255	***28232730155***	***-128905766325***
Client	52854	**6580188642**	**30847104000**
Conf	52854	**6580188642**	**30847104000**

This is in many ways the logical extreme of the "super-shingle" approach proposed by Broder *et. al.* [2], where the original shingle values are reshingled in groups of ten. The resulting much smaller table can be compared more quickly, but of course does not provide a way to detect similar documents. To provide an analogous number, we computed the shingles signature, by just adding up the shingles, shown in the last column of Table 5.

We note that while individual strings will usually have unique hash codes, there is a somewhat larger probability that the sum of a series of hash codes will be less unique. However, the probability of these collisions is small enough that these document signatures remain quite useful. Further, it is even less likely that documents with accidentally identical signatures would be returned from a query if they were not the same document.

To summarize, these document signatures simply provide a shorthand method of representing the top terms in documents, so that they can be compared very rapidly. The actual technique for comparison is essentially the same as in section 5.1.

7. QUERY 2 – SMALLER DOCUMENTS

In a second series of experiments, we issued a more focused query "Program ViaVoice in Java," and were able to retrieve 47 of the top 50 returned documents. Since many of these had the same filename, we carefully renamed them when we saved the local copies for analysis.

Since all of these documents were of modest size (the top one was 75 K) we found that we could perform the entire analysis on the documents fast enough that it could be carried out more or less in real time in response to a query.

The term signature results included 8 pairs of identical documents as measured by size and the signature we described above. In addition, the results contained the 13 very similar documents shown in Table 6.

Table 6 –Very similar documents returned by Query 2

Url	Size	Textract Signature	Shingle signature
News11	37788	-9902679640	*-2788203200*
News9	38100	**-9692972733**	**-10719056625**
News5	38383	-11688727862	*-2788203200*
News12	38604	**-9692972733**	**-10719056625**
News8	38727	-9921821918	-1343956500
News2	39389	**-9692972733**	**-2654568325**
News6	39389	**-9692972733**	**-2654568325**
News3	39465	**-9692972733**	***-4657627900***
News4	39465	**-9692972733**	***-4657627900***
News0	39580	-5116033555	*-3060800550*
News1	39580	-8166342237	*-3060800550*
News10	40537	-11188846377	**-5207686200**
News7	40537	-12715476873	**-5207686200**

The documents in Table 6 are all very similar, since they differ in only 1 or 2 terms out of 47, and all have similar sizes. Based on size alone, you would identify only 4 pairs of identical documents. However, all of these are detected as similar based on the fact that they contain the same terms. In addition, it is significant that six of these documents have identical signatures (shown in boldface) even though they are of four different sizes. This shows the power of the signature method for rapid identification of documents.

By contrast, the shingles signatures identified 6 pairs of identical documents, but did not recognize that 4 of these pairs were effectively identical as the text signatures did.

8. FINDING SIMILAR DOCUMENTS

In the foregoing, we have discussed the problem of finding very similar documents, in most cases so similar that only one of them need be returned from a search at all.

There is another set of problems to solve, however, relating to documents that are similar because they exist in a number of revisions. In order to test this algorithm for this problem, we determined that we should relax the restrictions regarding the percentage of terms that could be different, and the size differences we would allow between documents we compare.

In the first experiment, we collected 12 documents about IBM financial and banking products and services from the ibm.com web site, so that they would all have a relatively similar vocabulary, and one document which was a press release about IBM's performance in a supercomputer comparison. We expected that this last document would have a markedly different vocabulary from the others.

Then we took this last supercomputer document and cut out a 2533 byte segment comprising the main news story without any of the surrounding HTML formatting, and pasted it into each of the other financial documents. We then ran Textract and indexed the terms per document as described above and ran the same experiment on document similarity, where we changed the SQL query to allow the fraction of different terms to be as high as 0.5. This query identified every document pair correctly and did not find any pairs of documents similar except those consisting of an original and the same document with inserted text.

Table 7 shows the fraction of terms that differ between the original document and the same document when the news release text is inserted. The fractional text differences vary between 0.01 and 0.157. However, here shingles did not do nearly as well, with only 5 of the 12 document pairs being identified as similar.

We concluded that documents that had less than 20% of the terms different were likely to represent documents that were related and contained much the same text. In fact, we found it quite encouraging that this method identified every such document correctly and returned no false positives. (A false positive would be a difference in terms of less than 20% in documents that were in fact different.) In other words the precision and recall for the text method were 100%, while for the shingles method, the recall was less than 50%.

Table 7 – Fractional differences in terms in financial documents when an news release on supercomputers was added to each of them

#	Original url	Fraction of different terms with inserted news release	Fraction different using shingles
21	Folder1	0.100	0.28
38	Reuters	0.157	0.41
30	Nletter	0.06	0.06

20	Folder	0.117	0.86
17	Ebus3	0.076	0.84
16	Ebus1	0.055	1.00
35	Retaildel	0.096	0.01
27	Kookmin	0.054	0.03
1	24552	0.040	0.48
8	Building	0.040	0.03
14	Ebusmark	0.010	.26
33	RetailB	0.015	0

On the other hand, neither algorithm identified the short IBM press release document as being related to any of the others by containment, since it was relatively short, and contained fewer salient terms.

9. IMPLEMENTATION DETAILS

In these experiments, we ran the Textract text mining program on the collection of documents (around 50) returned from the query. Then we generated low-level DB2 table load files [19] from the Textract output and loaded the terms/document data into DB2. The IQ and frequency of the terms was determined from this collection. Thus, IQ would change somewhat based on the contents of the documents returned. A term that was highly salient in one document set might appear too frequently to be very selective in another set. However, we have eliminated much of this dependence by simply requiring that the IQ value be non-zero. It would in general be possible to maintain a vocabulary for a search system with IQs predetermined.

When all of the documents are relatively short, it is quite possible to do this more or less in real time. However, when longer documents make the mining processes too slow, it is necessary to index and mine the documents in advance and cache the results, just as you do with the document search indexes. When database comparisons of strings in very long documents can be slow, it is possible to just compare the top terms, for example the top 200 terms in each document. Further, we can store a numeric hash code for each term which can be compared more rapidly.

Finally, it is quite reasonable to store the document signature we describe as part of the database document table, so you can compare documents quickly.

In the course of these experiments, we varied the IQ threshold and the term frequency threshold. For various types of applications, these values may well need to be adjusted. However, it is important to note that the document signature is dependent on the number of terms you retrieve, and if you change your criteria, you will need to recompute these signatures.

In comparing documents for close similarity as we did in Query 1, we only considered documents that were within 10% of the size of the one we were comparing to, and only considered documents to be similar when the number of terms that were different was less than 10% of the total number of terms in the smaller document. In comparing documents that contained embedded additional material, we relaxed both of these criteria to 50%, with little performance penalty.

9.1 The Shingles Implementation

For PDF documents, we used the Acrobat Gemini plugin to convert them to HTML. For each HTML document, we extracted the pure text using the Java HTML classes, or when that was unsuccessful, saving the files as text from Internet Explorer. We then converted the entire document to lower case. The shingles were calculated using a Java program by using a moving 10-word window, which computes the Java String hashcode for each 10 word token. The set of hash codes were exported into a DB2 load file, loaded into a DB2 table, and the unique values extracted into a second table. For longer documents, we limited the total number of shingles *s* generated to those where

$s \, Mod \, 25 = 0$

as the authors proposed. For very short documents we also repeated the comparisons using all the shingles, with no significant change in the results.

10. SUMMARY AND CONCLUSIONS

We define similar documents as ones that have essentially the same sentences and paragraphs, but not necessarily in exactly the same order. We have found that we can accurately compute whether documents are similar by comparing the number of terms found using a phrase recognition program such as Textract. Using this technique, we can also give the user a list of the terms by which similar documents differ.

We further found that you can accurately recognize documents that have been revised to contain parts of other documents as still being closely related to the parent document. Finally, we described a novel document signature that you can use to make a rapid comparison between documents that are likely to be identical.

We contrast the success of this method with the shingles method as follows. For documents from the same source and file format, the techniques largely give the same results. However, when documents that originate in different file formats (such as PDF and HTML) are compared, the term-based method appears to be more successful. Further, while it would be possible to do some post-processing on translated PDF files before shingling them to try to eliminate differences, this amounts to the same sort of linguistic processing our technique already employs.

In the case of comparing documents with inserted text, the term method seems to be more successful, since its recall was much higher.

We further note that our term method uses only the salient terms to characterize documents, and these terms can appear in a different order and still provide the same characterization of the document. Further, phrase recognition programs such as Textract generally reduce the found terms to a root or "canonical" form, so that even if the terms appear in different variant forms in slightly edited versions of a document, they will be recognized as being the same root term and found to be identical. Finally, this method is insensitive to the addition of additional polishing sentences or the rearrangement of whole paragraphs in edited versions of a document.

This system has broad applicability in improving the results of searches of large document collections, whether the returned documents have been indexes for their term content in advance or not. It can also be used for rather sophisticated plagiarism detection, or as an adjunct in finding further documents of interest and grouping these documents for the user's convenience.

11. ACKNOWLEDGEMENTS

We thank Geoffrey Zweig of IBM Research for helpful discussions, and Robert Mack, Roy Byrd and Alan Marwick for their support.

12. REFERENCES

[1] Brown, Eric W. and Prager, John M., US Patent 05913208.

[2] Broder, Andrei Z, Glassman, Steven C., Manasse, Mark and Zweig, Geoffrey "Syntactic Clustering of the Web," *Proceedings of the Sixth WWW Conference.* Santa Clara, CA, 1997.

[3] Rabin, M. O., "Fingerprinting by random polynomials, " Center for Research in Computing Technology, Harvard University, Report TR-15-81, 1981.

[4] Bloomfield, Louis, University of Virginia, interviewed on NPR's *All Things Considered*, May 9, 2001. See www.plagiarism.phys.virginia.edu.

[5] Cooper, J. W. and Byrd, R J, "Lexical Navigation: Visually Prompted Query Refinement," ACM Digital Libraries Conference, Philadelphia, 1997.

[6] Cooper, James W. and Byrd, Roy J., OBIWAN – "A Visual Interface for Prompted Query Refinement," Proceedings of HICSS-31, Kona, Hawaii, 1998.

[7] Ravin, Y. and Wacholder, N. 1996, "Extracting Names from Natural-Language Text," IBM Research Report 20338.

[8] Justeson, J. S. and S. Katz "Technical terminology: some linguistic properties and an algorithm for identification in text." *Natural Language Engineering,* 1, 9-27, 1995.

[9] Byrd, R.J. and Ravin, Y. Identifying and Extracting Relations in Text. *Proceedings of NLDB 99*, Klagenfurt, Austria.

[10] Mnis-Textwise Labs, www.textwise.com. DR-LINK was developed at Syracuse University and is marketed by Textwise.

[11] Evans, D. K., Klavans, J. and Wacholder, N., "Document Processing with LinkIT," *Proc. Of the RIAO Conference,* Paris, France, 2000.

[12] InXight, Inc. www.inxight.com

[13] Neff, Mary S. and Cooper, James W. "Document Summarization for Active Markup," *in Proceedings of the 32nd Hawaii International Conference on System Sciences,* Wailea, HI, January, 1999.

[14] Cooper J.W. and Prager, John M. "Anti-Serendipity – Finding Useless Documents and Similar Documents," *Proceedings of the 33rd Hawaii International Conference on System Sciences, Maui, HI,* January, 2000.

[15] Cooper, J. W. "The Technology of Lexical Navigation," Workshop on Browsing Technology, *First Joint Conference on Digital Libraries,* Roanoke, VA, 2001.

[16] Cooper, J.W., Cesar, C., So, Edward, and Mack R. L., "Construction of an OO Framework for Text Mining," *OOPSLA,* Tampa Bay, 2001.

[17] Gemini plug-in for Adobe Acrobat Reader, Iceni Technology, Ltd, Norwich, England, www.iceni.com.

[18] Selker, T. and Burleson, W. "Context-aware Design and Interaction in Computer Systems," *IBM Systems Journal,* **39,** 891 (2000).

[19] Cooper, J W, "Loading Your Databases," *JavaPro,* May, 2000.