Document Ranking and the Vector-Space Model

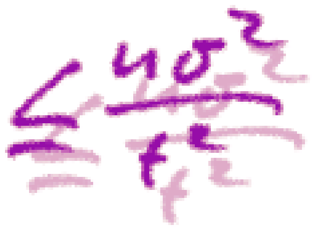
DIK L. LEE, Hong Kong University of Science and Technology
HUEI CHUANG, Information Dimensions
KENT SEAMONS, Transarc

Using several simplifications of the vector-space model for text retrieval queries, the authors seek the optimal balance between processing efficiency and retrieval effectiveness as expressed in relevant document rankings.

Efficient and effective text retrieval techniques are critical in managing the increasing amount of textual information available in electronic form. Yet text retrieval is a daunting task because it is difficult to extract the semantics of natural-language texts. Many problems must be resolved before natural-language processing techniques can be effectively applied to a large collection of texts.

Most existing text retrieval techniques rely on indexing keywords. Unfortunately, keywords or index terms alone cannot adequately capture the document contents, resulting in poor retrieval performance. Yet keyword indexing is widely used in commercial systems because it is still the most viable way by far to process large amounts of text. We face two main concerns when indexing text files.

0740-7459/97/$10.00 © 1997 IEEE



♦ *How do we identify index terms?* Index terms can range from single terms (with common words such as "the," "and," "to," and so on removed), to noun phrases, to subject identifiers derived from syntactic and semantic analysis.

Many systems, especially commercial ones, seldom make any performance evaluation data available.

♦ *After we identify index terms, how do we tell from them if a document matches a query?* In the Boolean model, a match is based on the satisfaction of a document's index terms to the Boolean expression given by the user, whereas the statistical model is based on the similarity between the statistical properties of the document and the query.

We focus here on the second issue. In particular, we examine several retrieval techniques that rank the result based on some similarity measure between the documents and the query. In the past, document ranking was supported only in research prototypes. Recently, more and more systems have begun supporting this feature because users demand easy-to-use query languages and it helps sort out the most important information for them.

Users are now exposed to document ranking features in a variety of domains through systems such as wide-area information systems (WAIS), World Wide Web robots, and online documentation systems. Unfortunately, these systems rarely address the core issue: the quality of the ranked output. Many systems, especially commercial ones, seldom make any performance evaluation data available. As a result, users often make statements like "system A is better than system B, because A supports ranking whereas B does not." Such statements are more deceptive than useful, because every system produces *a* ranking—even a system that simply returns a random set of documents can be said to be ranking them. We must determine how good a system's ranking procedures are before drawing conclusions regarding its value.

To establish a procedure for making such a determination, we first assumed that index terms are single words extracted from documents, then explored several ways to implement document ranking based on the vector-space model, which has been widely studied in the information-retrieval research community.[1] We experimented with several simplified implementations that reduce the complexity of the full vector-space model, focusing on relevance feedback and the development of methods for deriving terms from a relevant document that allow expansion of the original query.

The experiments we performed were all run on BasisPlus, a commercial document management system extended with ranking and relevance feedback.[2] BasisPlus provides a convenient platform for implementing and testing new algorithms. However, the results we present here reflect the algorithms being used, not the specific experimental platform upon which they are implemented. Similar studies have been reported elsewhere and their results are consistent with the observations made in this paper.[3,4] Compared to these others, our study includes new retrieval and feedback algorithms and a larger document collection.

RETRIEVAL EFFECTIVENESS

The conventional way of measuring the quality of the results returned by a system in response to a query is to use *precision* and *recall*. Precision is the number of relevant documents retrieved divided by the total number of documents retrieved. Recall is the number of relevant documents retrieved divided by the total number of relevant documents.

Ideally, the recall and precision should both be equal to one, meaning that the system returns all relevant documents without introducing any irrelevant documents in the result set. Unfortunately, this is impossible to achieve in practice. If we try to improve recall (by adding more disjunctive terms to the query, for example), precision suffers; likewise, we can only improve precision at the expense of recall. Furthermore, there is often a tradeoff between retrieval effectiveness and computing cost. As the technology moves from keyword matching to statistical ranking to natural-language processing, computing cost increases exponentially.

Statistical model. In the statistically based vector-space model, a document is conceptually represented by a vector of keywords extracted from the document, with associated weights representing the importance of the keywords in the document and within the whole document collection; likewise, a query is modeled as a list of keywords with associated weights representing the importance of the keywords in the query.

The weight of a term in a document vector can be determined in many ways. A common approach uses the so-called $tf \times idf$ method, in which the weight of a term is determined by two factors: how often the term j occurs in the document i (the term frequency $tf_{i,j}$) and how often it occurs in the whole document collection (the document frequency df_j). Precisely, the weight of a term j in document i is

$$w_{i,j} = tf_{i,j} \times idf_j = tf_{i,j} \times \log N/df_j,$$

where N is the number of documents in the document collection and *idf*

stands for the inverse document frequency. This method assigns high weights to terms that appear frequently in a small number of documents in the document set.

Once the term weights are determined, we need a ranking function to measure similarity between the query and document vectors. A common similarity measure, known as the *cosine measure*, determines the angle between the document vectors and the query vector when they are represented in a V-dimensional Euclidean space, where V is the vocabulary size.[1] Precisely, the similarity between a document D_i and a query Q is defined as

$$sim(Q,D_i) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\sum_{j=1}^{V} w_{Q,j}^2 \times \sum_{j=1}^{V} w_{i,j}^2}}$$

where $w_{Q,j}$ is the weight of term j in the query, and is defined in a similar way as $w_{i,j}$ (that is, $tf_{Q,j} \times idf_j$). The denominator in this equation, called the normalization factor, discards the effect of document lengths on document scores. Thus, a document containing $\{x, y, z\}$ will have exactly the same score as another document containing $\{x, x, y, y, z, z\}$ because these two document vectors have the same unit vector. We can debate whether this is reasonable or not, but when document lengths vary greatly, it makes sense to take them into account.

Feedback. An important feature of this model is *relevance feedback*, in which users judge the relevance of retrieved documents to their information need. According to the judgment, the system automatically adjusts the query vector and performs another round of retrieval, which will hopefully yield improved results.

One major advantage of the statistical model is that users can describe their information needs in natural language; the important keywords can be automatically extracted from the query in the same way keywords are extracted from documents. Thus, users are relieved from specifying complex Boolean expressions.

IMPLEMENTATION METHODS

Because the exact vector-space model is expensive to implement, we have developed a family of successively simpler approximations.

Method 1. The complexity of this method, the *full vector-space model*, depends on how we implement it. The document's vector representation is only conceptual. In practice, the full vector is rarely stored internally as is because it is long and sparse. Instead, document vectors are stored in an inverted file that can return the list of documents containing a given keyword and the accompanying frequency information. Besides, direct comparison between the vectors is slow because it would incur N vector comparisons. Vector comparison can be facilitated with an inverted file as follows.

```
for every query term $q$ in $Q$ do
    retrieve the postings list for $q$
        from the inverted file
    for each document $d$ indexed
        in the postings list do
        $score(d) = score(d) + tf_{d,q} \times idf_q$
    end
end
Normalize scores.
Sort documents according to
    normalized scores.
```

With an inverted file, the number of postings lists accessed equals the number of query terms. The computational cost is acceptable for queries of reasonable size. Unfortunately, the computation of the normalization factor is extremely expensive because the term $\sum_{j=1}^{V} w_{i,j}^2$ in the normalization factor requires access to *every* document term, not just the terms specified in the query. Nor can the normalization factor be precomputed under the $tf \times idf$ method, because every insertion and deletion on the document collection would change idf and thus the precomputed normalization factors.

Method 2. For this second method, to approximate the effect of normalization we use instead the square root of the number of terms in a document as the normalization factor. While this still favors long documents, the effect of document size is not as significant as it is without any normalization. This normalization factor is much easier to compute than the original one; also, precomputation is possible. With the approximation, the formula becomes

$$sim(Q,D_i) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\text{number of terms in } D_i}}.$$

Method 3. This method lets us further simplify the computation by simply dropping the normalization factor:

$$sim(Q,D_i) = \sum_{j=1}^{V} w_{Q,j} \times w_{i,j}\ .$$

That is, the document score equals the inner product of the document and query vectors. Instead of computing the angle between the document and query vectors, this formula computes the length of the projection of the doc-

The statistical model lets users describe their information needs in natural language.

TABLE 1
TEST COLLECTION CHARACTERISTICS

Collection name	Number of documents	Number of queries	Raw size (Mbytes)
CACM	3,204	64	1.4
CISI	1,460	112	1.2
CRAN	1,400	225	1.6
MED	1,033	30	1.0
TIME	425	83	1.5
TREC subset	10,000	25	25.9

```
Narrative description:

A relevant document will identify an information retrieval
system, identify the company or person marketing the system,
and identify some of the characteristics of the system.

Concept terms:

information retrieval system, storage, database, data, query.
```

Figure 1. *Query 14, a sample query from the TREC collection.*

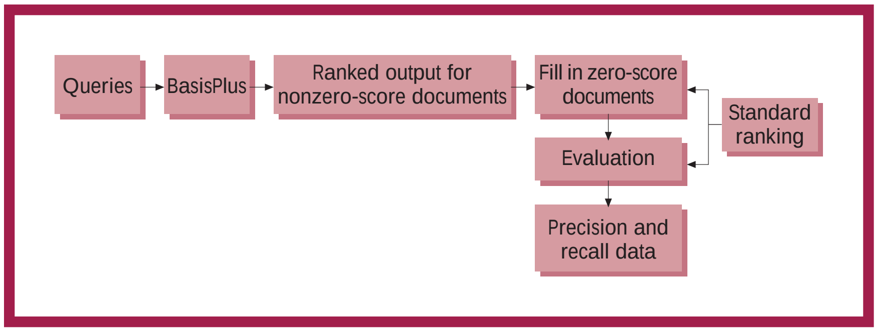

Queries
BasisPlus
Ranked output for nonzero-score documents
Fill in zero-score documents
Standard ranking
Evaluation
Precision and recall data

Figure 2. *Query evaluation process. Even zero-score documents are used to calculate precision and recall.*

ument vector onto the query vector. It is quite clear that the document score is directly proportional to the length of the document vector.

Method 4. This method only makes use of term frequencies in the calculation and ignores *idf*. It simplifies the computation as well as saving the file structure needed for storing the *df* values.

$$sim(Q, D_i) = \sum_{j=1}^{V} w_{Q,j} \times tf_{i,j}.$$

Method 5. This method ignores the term frequency (*tf*) information but retains the *idf* values in determining term weights:

$$sim(Q, D_i) = \sum_{j=1}^{V} w_{Q,j} \times w_{i,j},$$

$$\text{where } w_{Q,j} = \begin{cases} 1 & j \in Q \\ 0 & otherwise \end{cases}$$

$$\text{and } w_{i,j} = \begin{cases} idf_j & j \in D_i \\ 0 & otherwise. \end{cases}$$

The *idf* values have the same effect as before. That is, they diminish the significance of words that appear in a large number of documents.

Method 6. This method is the simplest in the family. It ignores both *tf* and *idf* values and therefore measures the number of common terms in the document and query vectors.

$$sim(Q, D_i) = \sum_{j=1}^{V} w_{Q,j} \times w_{i,j},$$

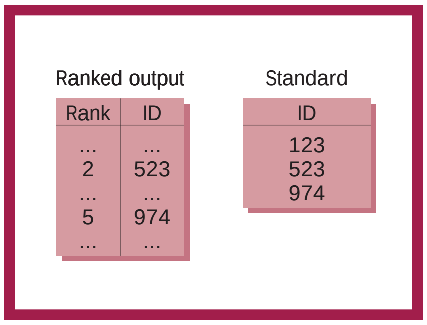
$$\text{where } w_{Q,j} = \begin{cases} 1 & j \in Q \\ 0 & otherwise \end{cases} \text{ and}$$

$$w_{i,j} = \begin{cases} 1 & j \in D_i \\ 0 & otherwise. \end{cases}$$

PERFORMANCE EVALUATION

The methods we've described obviously have different computational costs. To find their retrieval effectiveness, we obtain the precision and recall of these methods using a set of test collections. Table 1 summarizes the characteristics of the document collections. The CACM, CISI, CRAN, MED, and TIME collections are available with the Smart system developed at Cornell University.[5] Although small in size, these collections have been widely used by researchers for evaluating retrieval effectiveness. The TREC (Text Retrieval Conference) subset contains 10,000 *Wall Street Journal* articles extracted from the test collection used in the first Text Retrieval Conference, sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency to develop a comprehensive testbed.[6]

Each collection has a standard set of queries and an accompanying relevance judgment for each query. The TREC collection has 25 standard queries, each of which has a narrative description of the information needed and a number of concept terms identified by human experts based on the narrative



Ranked output

Rank	ID
...	...
2	523
...	...
5	974
...	...

Standard

ID
123
523
974

Figure 3. *Computing precision and recall values. Each document is ranked according to relevance. If a query does not retrieve a document (123 in this example), it receives the lowest ranking possible.*

description. Figure 1 shows a sample query. The concept terms are given by human experts and do not necessarily appear in the narrative text. CACM, CISI, CRAN, and MED provide only natural-language queries.

Query evaluation. Figure 2 outlines the basic evaluation process. This process is used to evaluate all six ranking methods: each method is implemented on BasisPlus and its precision and recall obtained. Most systems, including BasisPlus, do not return documents with zero scores. However, some of the zero-score documents may be relevant to the queries. Therefore, their ranks are required for calculating precision and recall. In the experiments, we assume the worst scenario: all relevant documents with zero scores are assigned the lowest possible ranks. For example, if there are *n* documents in the collection, relevant documents with zero scores are ranked as *n*, *n*-1, *n*-2, ..., and so on. The other alternatives are to rank zero-score documents randomly or simply by document IDs, but we do not recommend these methods because of the small chance that zero-score documents will receive high ranks.

The example in Figure 3 shows how the system calculates precision and recall values given the relevance judgment for a query. The retrieved document set is examined from the top. When the first relevant document is reached, the system calculates precision and recall. In the example, the precision would be 50 percent (one of the two retrieved documents is relevant) and the recall would be 33 percent (one of the three relevant documents is retrieved). Likewise, when the second relevant document is encountered (document 974 with rank 5), the precision and recall rise to 40 percent and 66 percent, respectively. If document 123 is not retrieved at all, it is assigned the lowest possible rank. For the CACM collection, the last rank would be 3,204. Therefore, the precision is 0.094 percent, whereas the recall is 100 percent. The precision values are then interpolated to obtain precision values at each recall value between 0 and 1 for every 0.05 increment, inclusively. The precision values for all queries at each recall point are then averaged to obtain the final precision–recall graph. For clarity, the precision–recall graph is further reduced to a single average precision over the 21 recall points, as shown in Figure 4. Interested readers may contact us to obtain the individual precision–recall graphs.[7]

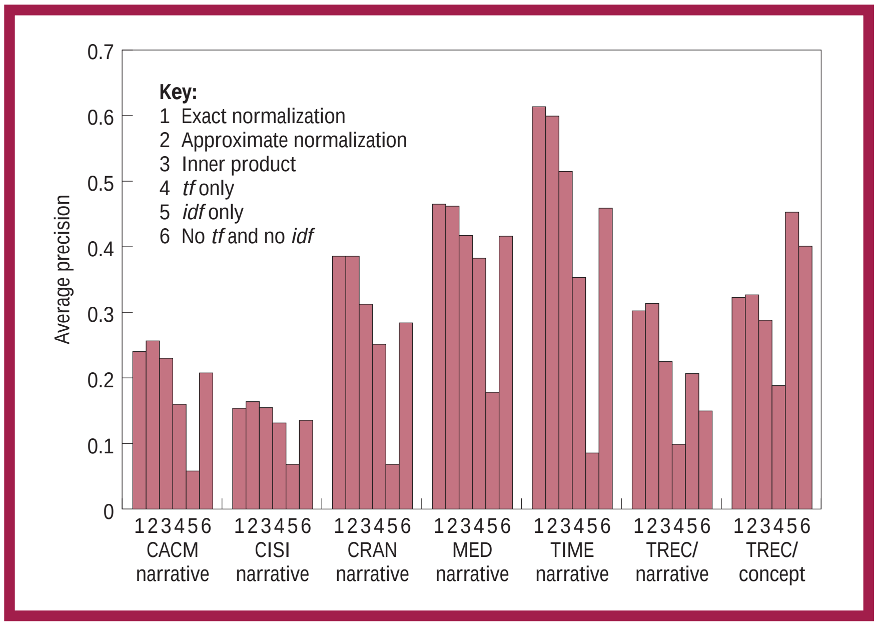


Figure 4. *Average precision for the six ranking methods on each of the seven sample document collections.*

Comparing methods. We intend the comparison shown in Figure 4 to hint at our methods' relative performance and hence their order of merit. The experimental results reflect system performance only at a particular operating point. For example, document collections (especially older ones) are built mostly based on availability of the documents; the collection developers typically capture queries over a narrow period of time. The TREC collection is better than others in this respect, but still doesn't represent every aspect of a live operational environment. Furthermore, when taking averages and interpolations on the experimental data, some information is inevitably lost. However, this loss won't affect the conclusion if a weak interpretation of the results is taken. Because of the nonrandomness of the system parameters, the applicability of statistical-significance tests is doubtful.[8,9] Thus, the procedure we have outlined remains a common method for evaluating retrieval effectiveness.

We have made the following observations regarding the different simpli-

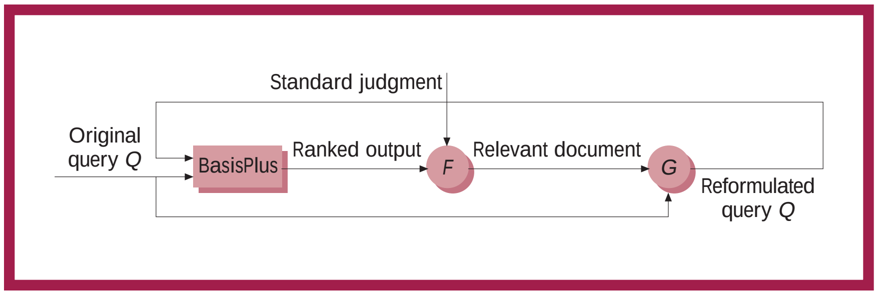


Figure 5. *Relevance feedback process using standard relevance judgment from the document sets.*

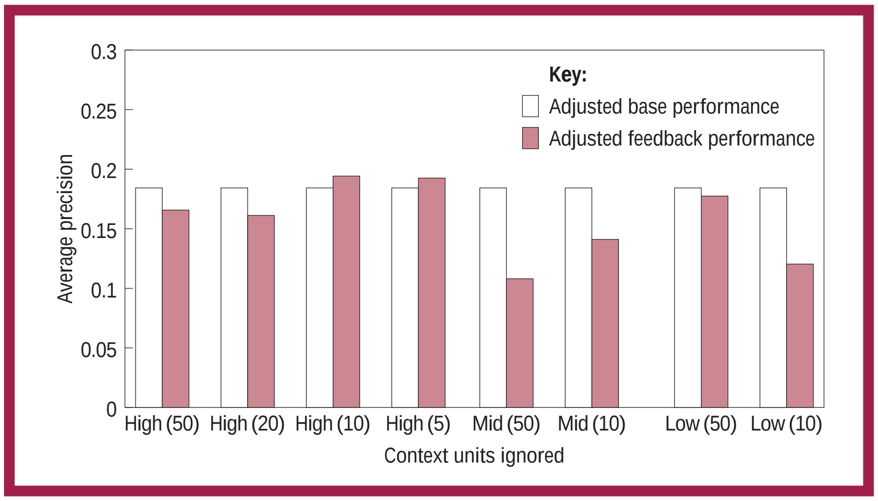


Figure 6. *Feedback performance with context units ignored. The Mid and Low options actually degrade performance.*

fications of the space-vector model.

♦ The approximate normalization factor works just as well as the exact normalization factor. In fact, it is better than the exact normalization factor in four of the seven collections and almost tied with it in two of the remaining three. Only in one case, the TIME collection, is approximate normalization slightly worse than the exact normalization factor. This result indicates that vector lengths should not be completely discarded in calculating document scores.

♦ The inner product (no normalization) is not as good as the first two methods, but provides a reasonable compromise given that it does not require explicit computation and storage of the normalization factor.

♦ The last three methods ignore some or all of the frequency information. In general, this group is not as good as the first three methods for natural-language queries. In particular, method 4, which ignores *idf* without ignoring *tf*, is consistently worse than any of the first three methods.

♦ Methods 5 and 6, which try to measure the overlap between the document and query terms, perform *extremely* well for the concept query, although they perform relatively poorly for natural-language queries. This could be attributed to the concept terms being very precise content descriptors. Thus, the appearance of a concept term in a document will almost always reveal the document's relevance, regardless of the term frequency information.

♦ For concept queries, method 5, which takes *idf* into account, is better than method 6. This confirms that a concept that appears in many documents should be given a small weight.

♦ However, method 5 performs *extremely* poorly for natural-language queries. This may be attributed to the inconsistent use of frequency information. In other words, *idf* and *tf* each represent only half of the equation in computing term weights; therefore neither should be used alone. The poor performance of method 4 supports this observation.

As to which method we recommend using, our experiments provide strong evidence that method 2, using the approximate normalization factor, is best for natural-language queries, whereas method 5 (closely followed by method 6) works best for concept queries. This suggests that for a general information retrieval system, both methods should be supported.

The inner product should be used only for compatibility reasons, since most Boolean systems, including BasisPlus, do not keep enough information for computing the normalization factor easily. Method 1 is infeasible for implementation as explained before, while there is no particular situation in which method 4 would excel.

RELEVANCE FEEDBACK

Relevance feedback is an important strength of the vector-space model. The idea is that queries specified by end users typically fail to describe completely what those users want. Thus, typical queries miss many relevant documents. However, if the user can identify some retrieved documents as relevant, then the system can use this information to reformulate the original query into a new one that may capture some of the concepts not explicitly specified in the original query. For example, if the original query contains "commodity price" and the user marks a document containing "gold price" as relevant, the feedback process may include the term "gold price" in the reformulated query. This might retrieve articles on "gold futures, "gold index," and so on.

The main problem with relevance feedback is that including a large piece of text indiscriminately in the original query will adversely affect effectiveness. Doing so also incurs the typical high cost of processing long queries. Continuing with our example, if the relevant document contains "gold price in the London market," then "London" and "market" may also be includ-

ed in the reformulated query, resulting in retrieval of irrelevant articles mentioning "London." Therefore, criteria must be carefully set up to select terms from the relevant texts for query expansion. We define three methods—High, Mid, and Low—that select high-, medium-, and low-frequency terms, respectively, from the relevant text for query expansion.

Because BasisPlus supports the notion of context units, which are user-defined logical units of a document (for example, a sentence, a paragraph, or even the entire document), we can ask the system to select terms only from context units containing at least one hit—a match between a document term and a query term—and ignore the others. This is called the Context method.

The terms in the selected context units will then be filtered according to the frequency criterion specified by the user as High, Mid, or Low. When context units are used, we further allow one more way of selecting terms: terms in the neighborhood of the hits in the relevant text (denoted as the Hits method). Thus, combinations can be specified as follows:

No Context and High, Mid, or Low
Context and High, Mid, Low, or Hits

For each option, the number of terms to be selected can be specified. For example, <Context, High(10)> means the 10 most frequent terms from the context units containing at least one hit will be included in the original query, whereas <Hits(10)> includes 10 terms from the left and right neighborhood of a hit.

Because this is a large array of options, we conducted an experiment to find out which ones work better. Since relevance feedback involves user intervention and it is infeasible to run many experiments manually, we simulated the feedback action in the experiment. Figure 5 shows that after a retrieval is performed process F takes as input the relevance judgment provided with the queries, scans down the retrieved list, and pulls out the first relevant document, which is then selected as the feedback document in the feedback process. This simulates the user's feedback action. That is, if the user is indeed given the ranked output and is patient enough to examine each document from the beginning, he or she would have identified the same document as selected in the experiment, assuming that the standard judgment is complete and correct and that the user does not make an erroneous judgment.

After the relevant document is identified, process G combines the terms from the original query and those from the relevant document to form a new query:

$$Q' = Q + aR - bI,$$

where $0 \le \alpha \le 1$, $0 \le \beta \le 1$, R is the set of relevant documents, and I is the set of irrelevant documents as judged by the user. The effect of aR is to expand the original query with terms from the relevant documents and to increase the weights of those terms already in the original query. bI does exactly the opposite for irrelevant documents.

In the experiment, we set a to 1, b to 0, and $|R| = 1$. This is equivalent to asking the user to identify one relevant document and to ignore the irrelevant ones.

PERFORMANCE EVALUATION

We ran three sets of experiments to evaluate the effectiveness of each feedback method. The first set ignores context units; the second set uses sentences as context units. The last set uses paragraphs as context units but it only uses the Hits and High options, because the first two experiment sets showed that Mid and Low are not good term selection methods. These experiments use only the TREC subset because the documents in it are long enough to allow the range of tests performed.

The documents identified as relevant during the feedback process must be discarded when evaluating relevance performance. This is crucial and must be done or these documents will be ranked very high—if not highest—in subsequent iterations, either because a large portion of the terms in them are included in the reformulated query or because the system automatically puts them at the top without further processing. These documents, if not removed when precision and recall are computed, will inflate the performance of the feedback query, since they are actually picked by the users and not the system.

Suppose that D_1 and D_2 are the document sets retrieved by the original and reformulated queries, respectively, that R is the standard set of relevance documents, and that D' is the set of documents identified by the user as relevant in the feedback process. The adjusted baseline performance is the precision and recall obtained based on $D_1 - D'$ and $R - D'$. The adjusted feedback performance is based on $D_2 - D'$ and $R - D'$. These two performance values indicate, respectively, how good the original and reformulated queries are in ranking the relevant documents in $R - D'$, and thus provide a common basis for comparison.

Queries specified by end users typically fail to describe completely what those users want.

From the results shown in Figure 6, we can see that the Mid and Low options actually make the feedback performance worse in all cases, whereas the High(10) and High(5) options improve performance. Mid and Low options produce poor results because the terms with low or middle frequen-

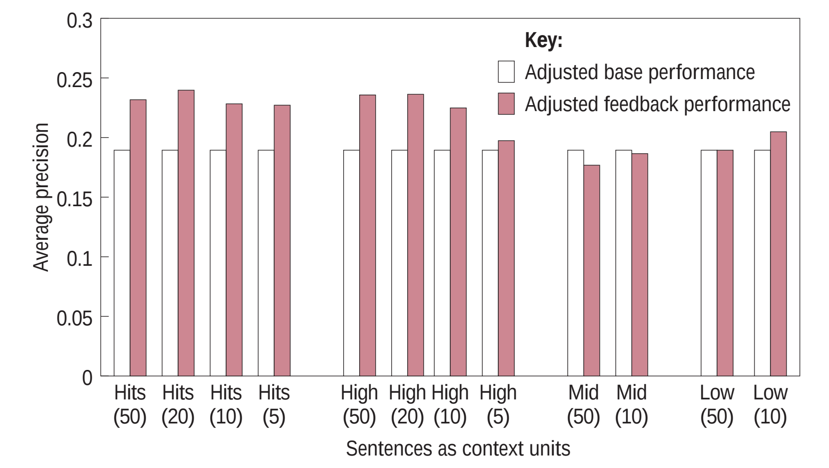


Figure 7. *Feedback performance using sentences as context units. Mid and Low options still cause poor performance. The Hits(20) and High(50) options offer the best performance.*

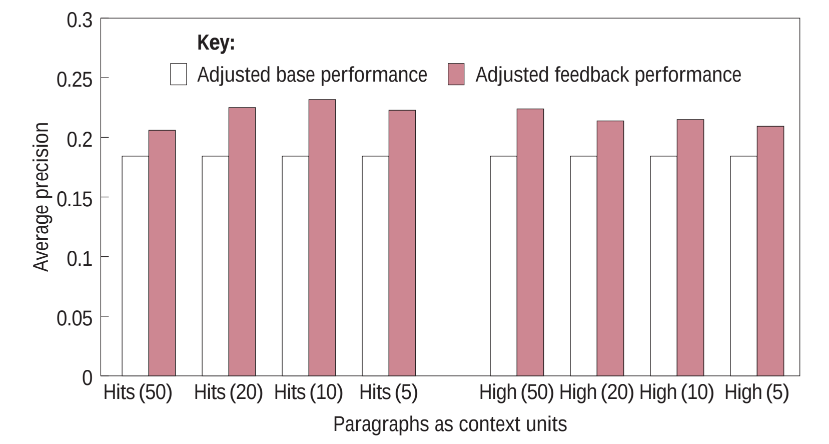


Figure 8. *Feedback performance using paragraphs as context units. The Hits(10) and High(50) options offer the best performance.*

cies are unlikely to represent the main concepts in a document. Thus, including these terms in the query will result in only irrelevant documents being retrieved. On the contrary, high-frequency terms tend to be important terms, so they are good candidates for expanding the original query. But too many of them would degrade performance, which argues against arbitrarily expanding the query with a large number of terms.[10]

We repeated the experiments using sentences and paragraphs as context units. For the former, we tested all methods, whereas we used only the High and Hits methods for the latter. The average precisions are shown in Figures 7 and 8. Figure 7 shows that the Mid and Low options still give poor feedback performance, which is consistent with our previous observation. However, the feedback performance shows significant improvement over the baseline performance for both the High and Hits options. The best case for Hits is Hits(20) and the best for High is High(50). The performance gain is as much as 25 percent over baseline. With context units used, more terms can be used to expand the query before the performance starts to degrade. This shows that context units containing one or more hits tend to contain words relevant to the search topic.

The results show that selecting terms from context units containing at least one hit is better than picking them from the entire document. Also, high-frequency terms and terms around hits are good indicators of the document contents. However, there is no overwhelming evidence to indicate which combination works best. In practice, the context unit is usually dictated by other aspects of the retrieval system, so the only decision for the user to make is the choice between the High and Hits options.

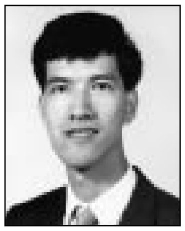
We have described a family of six implementations for the vector-space model and their retrieval effectiveness. For concept queries, we find it best to use only inverse document frequencies while ignoring term frequencies, but for natural-language queries both frequencies are needed—ignoring either one will produce poor results. Furthermore, the approximate normalization factor we suggest here gives the best performance and is computationally efficient.

We have found that high-frequency terms or terms in the neighborhood of hits selected from context units containing at least one hit give the best feedback performance. In some cases, the gain in average precision is as much as 20–25 percent. On the other hand, although the data shows slight improvement in using terms around hits over high-frequency terms, the evidence is not overwhelming.

To develop an efficient and effective retrieval system, many implementation issues must be considered. For example, index structures suitable for terabytes of data must be developed; query processing methods that search the index based on the semantics and relative importance of the query terms can potentially reduce the processing time dramatically. Furthermore, query models that treat a query as a semantic hierarchy of concepts, not merely as a bunch of independent keywords, hold great promise for improving retrieval effectiveness and are increasingly used in information filtering for Internet-based information dissemination systems. ◆

REFERENCES

1. G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.
2. "The Complete FIND Handbook," The BASISplus Document Set, Information Dimensions, Dublin, Ohio, Nov. 1992.
3. G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management*, Vol. 24, No. 5, 1988, pp. 513-523.
4. G. Salton and C. Buckley, "Improving Retrieval Performance by Relevance Feedback," *J. Amer. Soc. for Information Science*, Vol. 41, No. 4, 1990, pp. 288-297.
5. C. Buckley, "Implementation of the SMART Information Retrieval System," TR 85-686, Cornell Univ., Ithaca, N.Y., May 1985.
6. D. Harman, "Overview of the First Text Retrieval Conference," *Proc. 16th Int'l ACM/SIGIR Conf. Research and Development in Information Retrieval*, Pittsburgh, June 1993, pp. 36-47.
7. D.L. Lee, "Document Ranking in BASISplus," *Information Dimensions*, Dublin, Ohio, 1993.
8. E.M. Keen, "Presenting Results of Experimental Retrieval Comparisons," *Information Processing and Management*, Vol. 28, No. 4, 1992, pp. 491-502.
9. *Information Retrieval Experiment*, K. Sparck-Jones, ed., Butterworths, London, 1981.
10. C. Stanfill and B. Kahle, "Parallel Free-Text Search on the Connection Machine System," *Comm. ACM*, Dec. 1986, pp. 1229-1239.



Dik Lun Lee is Reader of computer science at the Hong Kong University of Science and Technology. Prior to this he was associate professor of computer and information science at Ohio State University. His research interests include document retrieval, object-oriented systems, information discovery and management, and mobile computing. He has served as the editor and guest editor for several technical publications and has been an ACM lecturer.

Lee received a BS in electronics from the Chinese University of Hong Kong and an MS and PhD in computer science from the University of Toronto. He is a member of the IEEE, IEEE Computer Society, and ACM.

Huei Chuang is a technical staff member of Information Dimensions, where he developed the BasisPlus system.

Chuang received a PhD in computer science from Ohio State University.

Kent E. Seamons is a member of the group conducting research in wide-area information systems at Transarc Corporation. He has held technical positions at Information Dimensions and the National Center for Supercomputing Applications. His research interests include database systems, information retrieval, and parallel I/O.

Seamons received a BS in computer science from Brigham Young University and a PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the IEEE and ACM.

Address questions about this article to Lee at The Hong Kong University of Science & Technology, Department of Computer Science, Clear Water Bay, Hong Kong; dlee@cs.ust.hk.