# Effectiveness of Document Ranking and Relevance Feedback Techniques

Dik L. Lee
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong

Huei Chuang and Kent Seamons
Information Dimensions, Inc.
Dublin, Ohio
USA

January, 1996

**Abstract**

This paper discusses key issues in document ranking techniques based on the vector-space model. Different simplifications to the vector-space model and their impacts on retrieval effectiveness are described. We further discuss relevance feedback and several ways of modifying the user query based on feedback information. We show the experimental results for the retrieval effectiveness of the techniques and explain the experiment design for obtaining the results. In addition to existing methods, this paper introduces new methods for implementing and document ranking and relevance feedback. Moreover, practitioners will find this paper a useful guide for accessing the technology in this increasingly important area.

**Keywords and Phrases:** information retrieval, document retrieval, relevance feedback, retrieval effectiveness, evaluation

## 1 Introduction

Efficient and effective text retrieval techniques are critical in managing an increasing amount of textual information available in electronic form. Unfortunately, text retrieval is a difficult task because it is very difficult to extract the semantics of natural language texts. Many problems have to 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 from the text. Unfortunately, keywords, or index terms, alone can hardly capture the document contents, thus resulting in poor retrieval performance. Keyword indexing is widely used in commercial systems, since it is by far the most viable way for processing large amount of text.

There are two main concerns in indexing.

- How to identify index terms? Index terms can range from single terms (with common words removed), to noun phrases, to subject identifier derived from syntactic and semantic analysis.
- After index terms are identified, how do we tell from the index terms if a document matches a query? In the Boolean model, 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.

This paper focuses on the latter issue. In particular, we discuss 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. Recent years have seen more and more systems supporting this feature. Users are 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, the major issue, that is, how good the ranked output is, is rarely addressed. Many systems, especially commercial systems, seldom have any performance evaluation data available. As a result, statements like "A is better than B, because A supports ranking whereas B does not" are often made. These statements are more deceptive than useful, because every system produces *a* ranking — a system which simply returns a random set of documents can be said to be performing document ranking. The important question of how good the ranking is must be answered before any conclusion on the value of the ranking can be drawn.

This paper examines several ways of implementing document ranking based on the vector-space model. We investigate several simplified implementations which reduce the complexity of the full vector-space model. We study relevance feedback and several methods for deriving terms from a relevant document for expanding the original query. Experiments reported in this paper were all run on BASISplus, a commercial document management system from Information Dimensions, Inc., extended with ranking and relevance feedback [3]. BASISplus provides a convenient platform for implementing and testing new algorithms. However, the results presented in this paper reflect the algorithms being used, not the experimental platform upon which the algorithms are implemented. Similar studies have been reported in the literature [6,7]. Our study includes new retrieval and feedback algorithms and a larger document collection in the experiments. In addition to performance, we describe the evaluation and experimental design.

Section 2 describes the criteria for measuring retrieval effectiveness. An overview of the vector-space model is given in Section 3. Section 4 presents a family of six implementations of the vector-space model and their performance. Relevance feedback and the performance of several methods for query expansion are described in Section 5.

## 2 Retrieval Effectiveness

The conventional way of measuring retrieval effectiveness, or the quality of the results returned by a system in response to a query, is precision and recall defined as follows.

$$\text{recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}}$$

$$\text{precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of document retrieved}}.$$

Ideally, the recall and precision should both equal to one, meaning that the system returns all relevant documents without introducing any irrelevant documents in the result set. Unfortunately, this ideal performance is impossible to achieve in practice. In general, if we try to improve recall (e.g., by including more disjunctive terms into the query), precision will suffer, and vice versa. Furthermore, there is often a tradeoff between retrieval effectiveness and computing cost. As the

Table 1: Notations and definitions.

| | | |
|---|---|---|
| $N$ | $=$ | number of documents |
| $V$ | $=$ | vocabulary size |
| $df_j$ | $=$ | document frequency of term $j$ |
| | | number of documents containing term $j$ |
| $idf_j$ | $=$ | inverse document frequency of term $j$ |
| | | $\log_2 \frac{N}{df_j}$ |
| $tf_{i,j}$ | $=$ | frequency of term $j$ in document $i$ |
| $w_{i,j}$ | $=$ | weight of term $j$ in document $i$ |
| | | $tf_{i,j} \times idf_j$ |
| $w_{Q,j}$ | $=$ | weight of term $j$ in query $Q$ |

technology moves from keyword matching, to statistical ranking to natural language processing, computing cost increases exponentially.

# 3 Statistical Model

In the 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. To facilitate discussion, we summarize in Table 1 some of the notations and their definitions used in this section.

The weight of a term in a document vector can be determined in many ways. A common method is the so-called $tf \times idf$ method, where the weight of a term is determined by two factors: the number of occurrences of the term in the document (the term frequency $tf$) and the number of occurrences of the term in the whole document collection (the document frequency $df$). 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 which appear frequently in a small number of documents in the document set.

Once the term weights are determined, a ranking function is needed to measure the similarity between the query and document vectors. A common similarity measure, known as the cosine measure, measures the angle between the document vectors and the query vector when they are represented in an $V$-dimensional Euclidean space, where $V$ is the vocabulary size [8]. Precisely, the similarity between a document $D_i$ and a query $Q$ is defined as:

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

where $w_{Q,j}$ is the weight of term $j$ in the query, which is defined in a similar way as $w_{i,j}$ (i.e., $tf_{Q,j} \times idf_j$). The denominator in (1), called 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. Whether or not this is reasonable is debatable, but when document lengths vary greatly, it makes sense to take document lengths into account.

An important feature of this model is *relevance feedback*, in which the user judges the relevance of the retrieved documents to his/her 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.

# 4 Various Implementation Methods

In this section, we discuss the implementation of the vector-space model and several approximations. We will explain why the exact vector-space model is expensive to implement and then introduce a family of successively simpler approximations. We suggest a new normalization factor which is easy to compute and has comparable average precision as the exact vector-space model according to the performance evaluation shown at the end of this section.

### Method 1: Full vector-space model

The complexity of the full vector-space model depends on how it is implemented. The vector representation of a document is only conceptual. In practice, the full vector is rarely kept because it is long and sparse. Besides, direct comparison between the vectors is slow since 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 is equal to 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. Note that the normalization factor cannot 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: Approximate normalization factor

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 normalization factor still favors long documents, the effect of document size is not as significant as without any normalization. This normalization factor is much easier to compute than the original one; also, precomputation is possible with this normalization factor. 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: No normalization factor

To further simplify the computation, we can simply drop the normalization factor:

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

That is, the document score is equal to 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 document 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: Ignoring *idf*

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: Ignoring $tf$

This method ignores the term frequency information but retaining the *idf* values in determining term weights:

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

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

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

### Method 6: Ignoring both $tf$ and *idf*

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.

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

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

Table 2: Characteristics of the test collections.

| 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: A sample query.

### Performance Evaluation

It is clear that the methods described above have different computational costs. To find out their retrieval effectiveness, we obtain the precision and recall of these methods using a set of test collections. The characteristics of the document collections are summarized in Table 2. The CACM, CISI, CRAN, MED, and TIME collections are available with the SMART systems developed at Cornell University [1]. Although small in size, these collections have been widely used by researchers for evaluating retrieval effectiveness. The TREC subset is a set of 10,000 Wall Street Journal articles extracted from the test collection used in the first TREC (Text REtrieval Conference), which is sponsored by NIST and ARPA to develop a comprehensive testbed [2].

Each collection comes with a standard set of queries and the relevance judgement for each query. The TREC collection comes with 25 standard queries, each of which has a narrative description of the information need and a number of concept terms identified by human experts based on the narrative description. A sample query (query 14) is given in Figure 1. The concept terms are given by human experts and they do not necessarily appear in the narrative text. CACM, CISI, CRAN and MED provide only natural language queries.

Figure 2 outlines the basic evaluation process. 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, the worst scenario is assumed. That is, all relevant documents with zero scores are assigned the lowest possible ranks. For example, if there are $n$ documents in the collection, relevant documents

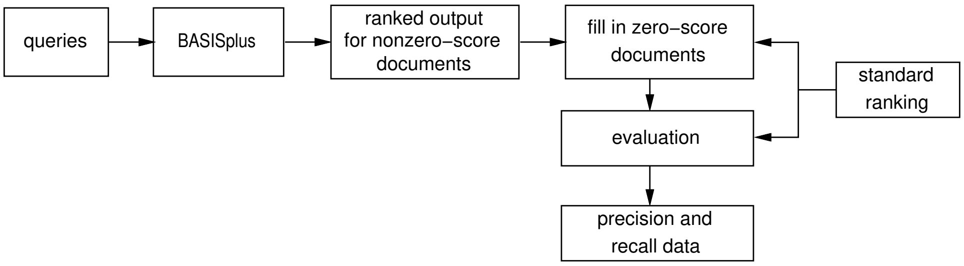


Figure 2: Evaluation process.

ranked output

| rank | ID |
|---|---|
| ... | ... |
| 2 | 523 |
| ... | ... |
| 5 | 974 |
| ... | ... |

standard

| ID |
|---|
| 123 |
| 523 |
| 974 |

Figure 3: Computing precision and recall values.

with zero scores are ranked as $n, n-1, n-2, ...$, etc. The other alternatives are to rank zero-score documents randomly or simply by document IDs, but these methods are not recommended since there is a small chance that zero-score documents will receive high ranks under these methods.

An example is shown in Figure 3 to illustrate how precision and recall values are calculated given the relevance judgement for a query. The retrieved document set is examined from the top. When the first relevant document is reached, the precision and recall are calculated at this point. In the example, the precision would be 50% (1 out of the 2 retrieved documents is relevant) and the recall would be 33% (1 out of the three relevant documents is retrieved). Likewise, when the second relevant document is encountered (document 974 with rank 5), the precision and recall are 40% and 66%, respectively. If document 123 is not retrieved at all (i.e., has zero score), it is assigned the lowest possible rank. For the CACM collection, the last rank would be 3,204. Therefore, the precision is 0.094%, whereas the recall is 100%. The precision values are then interpolated to obtain precision values at each recall value between 0 and 1 at 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 (see Figure 4). The individual precision-recall graphs can be found in a technical report obtainable from the authors [5].

It must be emphasized that the comparison shown in this and subsequent charts are meant to give a hint on the relative performance and hence the order of merit of the methods. The experimental results only reflect the system performance at a particular operating point. For instance, document collections (especially older ones) are built mostly based on availability of the documents and queries are captured typically by the collection developers over a narrow period of time. The TREC collection is better than others but it still doesn't represent every aspect

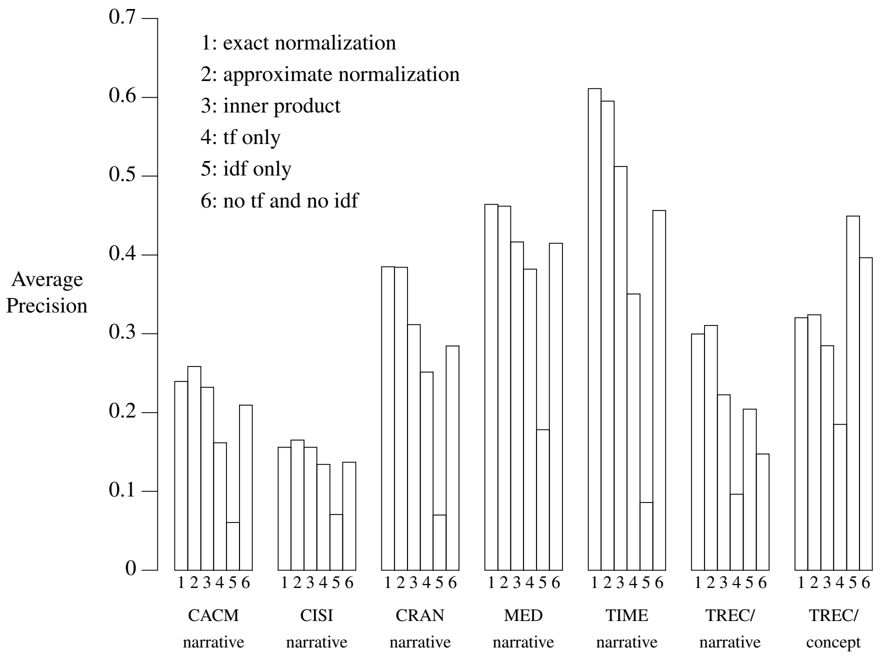


Figure 4: Average precision for every ranking method on every document collection.

of a life operational environment. Furthermore, when taking averages and interpolations on the experimental data, some information is inevitably lost. However, it 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 [4,9]. Thus, the procedure outlined above remains a common method for evaluating retrieval effectiveness.

The key observations are summarized as follows.

- The approximate normalization factor works just as good as the exact normalization factor. In fact, it is better than the exact normalization factor in four out of the seven collections and almost tied with it in two out of the seven collections. Only in one case (TIME) is it slightly worse than the exact normalization factor. This means that the vector lengths should not be completely discarded in calculating document scores.

- The inner product (i.e., no normalization) is not as good as the first two methods, but it is a reasonable compromise given that it does not require computing and storing the normalization factor explicitly.

- 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 (see below for concept 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 fact that the concept terms are very precise content descriptors. Thus, the appearance of a concept term in a document will almost always reveal the relevance of the document, regardless of the term frequency information.

- For concept queries, Method 5, which takes *idf* into account, is better than Method 6. This confirms the idea that a concept which 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 inconsistency in the use of frequency information. In other words, *idf* and $tf$ each is only half of the equation in computing terms weights and therefore should not be used alone. This observation is also supported by the poor performance in Method 4.

As a recommendation on which method to be used, there is strong evidence from the experiments that Method 2 (using approximate normalization factor) is the best for natural language query, whereas Method 5 (with Method 6 closely behind) is recommended 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 where Method 4 would excel.

## 5 Relevance Feedback

Relevance feedback is an important strength of the vector-space model. The idea behind relevance feedback is that queries specified by end users typically fail to describe completely what the users exactly want. Thus, many relevant documents will be missed. However, if the user is able to identify some retrieved documents as relevant, then this information can be used to reformulate the original query into a new one which may capture some of the concepts not explicitly specified in the original query. For instance, if the original query contains "commodity price" and the user marks a document containing "gold price" as relevant, then the feedback process may include the term "gold price" into the reformulated query, which may, ideally, retrieve articles on "gold future, "gold index," etc.

The main problem of relevance feedback is that including a large piece of text indiscriminately into the original query will adversely affect effectiveness, not to mention the high cost in processing long queries. Following the same example above, if the relevant document contains "gold price in the London market," then "London" and "market" may also be included into the reformulated query, resulting in retrieval of irrelevant articles mentioning "London." Therefore, criteria must be set up to select terms from the relevant texts for query expansion. We define three methods, namely, HIGH, MID and LOW, which, respectively, select terms with high frequency, middle frequency and low frequency from the relevant text for query expansion.

Since BASISplus supports the notion of context units, which are user defined logical units of a document (e.g., a sentence, a paragraph, or the entire document can be defined as a context unit), we can ask the system to select terms only from context units containing at least one hit (the method is denoted as CONTEXT) and ignore the others. The terms in the selected context units will then be filtered according to the frequency criterion specified by the user (i.e., HIGH, MID

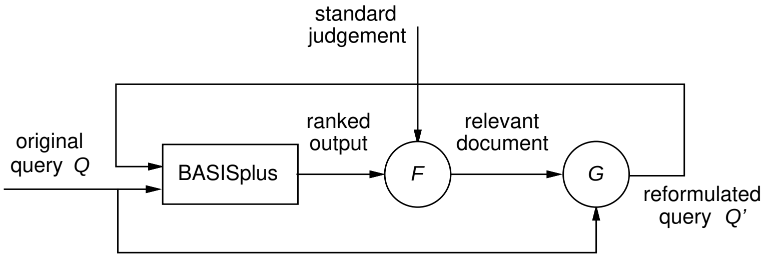


Figure 5: Relevance feedback.

or LOW). When context units are used, we further allow one more way of term selections: terms in the neighborhood of the hits — matches between a document term and a query term — in the relevant text (the method is denoted as HITS). Thus, the following combinations can be specified:

| | | | | |
|---|---|---|---|---|
| NO CONTEXT | HIGH | MID | LOW | N/A |
| CONTEXT | HIGH | MID | LOW | HITS |

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

Since a selection must be made from this large array of options either by the users or by the system, an experiment is conducted to find out which options are better. Since relevance feedback involves user intervention and it is infeasible to run a large number of experiments manually, the feedback action is simulated in the experiment. As shown in Figure 5, 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 process 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/she would have identified the exact 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. In general, the user can also specify which documents are irrelevant and a scaling factor could be used when combining terms.

$$Q' = Q + \alpha R - \beta I,$$

where $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$, $R$ is the set of relevant documents, and $I$ is the set of irrelevant documents as judged by the user. The effect of $\alpha R$ is to expand the original query with terms from the relevant documents and to increase the weights of those terms which are already in the original query. $\beta I$ does exactly the opposite for irrelevant documents. In the experiment, we set $\alpha$ to 1, $\beta$ to 0 and $|R| = 1$. This means that the user is asked to identify one relevant document and that irrelevant documents are ignored.

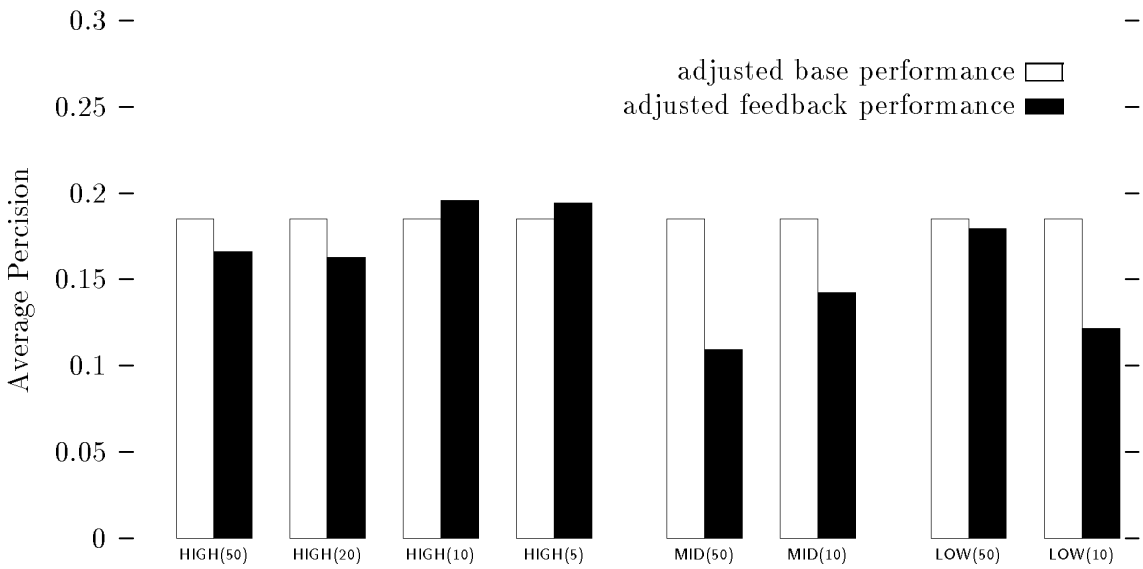


Figure 6: Feedback performance with context units ignored.

### Performance evaluation

Three sets of experiments are run. 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 HITS and HIGH, because the first two sets of experiments have shown that MID and LOW are not good term selection methods. In these experiments, only the TREC subset is used since the documents in it are long enough to allow the range of tests performed.

A crucial point must be noted when evaluating feedback performance. The documents selected by the user during the feedback process must be discarded when relevance performance is evaluated. This is because these documents will be ranked very high, if not the highest, in subsequent iterations, either because a large portion of the terms in these documents are included in the reformulated query or because the system automatically puts these documents 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 these documents are actually picked by the users not by the system. Suppose $D_1$ and $D_2$ are the document sets retrieved by the original and reformulated queries, respectively, $R$ is the standard set of relevance documents, and $D'$ 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.

From the results shown in Figures 6, it is clear that MID and LOW actually make the feedback performance worse in all cases, whereas improvement is observed for HIGH(10) and HIGH(5). The fact that MID and LOW are poor can be explained by the fact that terms with low or middle frequencies do not likely represent the main concepts in a document. Thus, including these terms in the query will only result in irrelevant documents being retrieved. On the contrary, high frequency

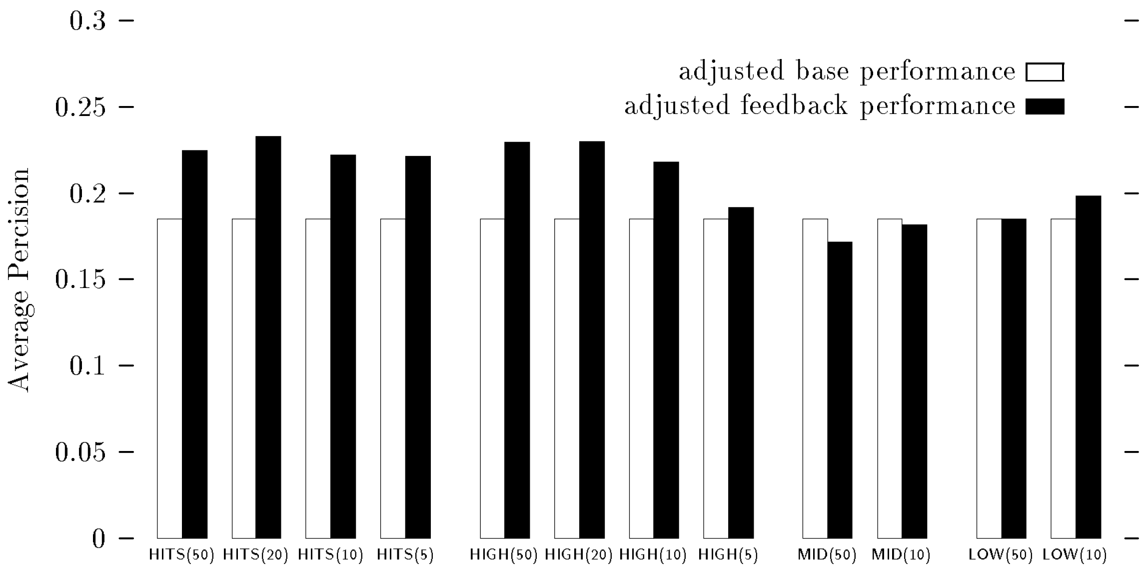


Figure 7: Feedback performance using sentences as context units.

terms tend to be important terms so they are good candidates for expanding the original query, but too many of them would degrade performance. The latter point argues against arbitrarily expanding the query with a large number of terms [10].

The experiments are repeated using sentences and paragraphs as context units. For the former, all four methods, namely, HIGH, MID, LOW, and HITS, are tested, whereas only HIGH and HITS are used for the latter. The average precisions are shown in Figures 7 and 8. We can observe from Fig. 7 that MID and LOW still give poor feedback performance, which is consistent with the previous observation. However, the feedback performance shows significant improvement over the baseline performance for both HIGH and HITS. The best case for HITS is HITS(20) and the best for HIGH is HIGH(50). The gain is as much as 25% over the baseline performance. With context units used, a larger number of 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 superior to picking them from the entire document and that high frequency terms and terms around hits are good indicators of the document contents. However, there is no overwhelming evidence showing which combination is the best. In practice, the context unit is usually dictated by other aspects of the retrieval system, so the only decision to make is to choose between HIGH and HITS.

# 6 Conclusion

The paper describes a family of six implementations for the vector-space model and their retrieval effectiveness. It is found that for concept queries it is 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

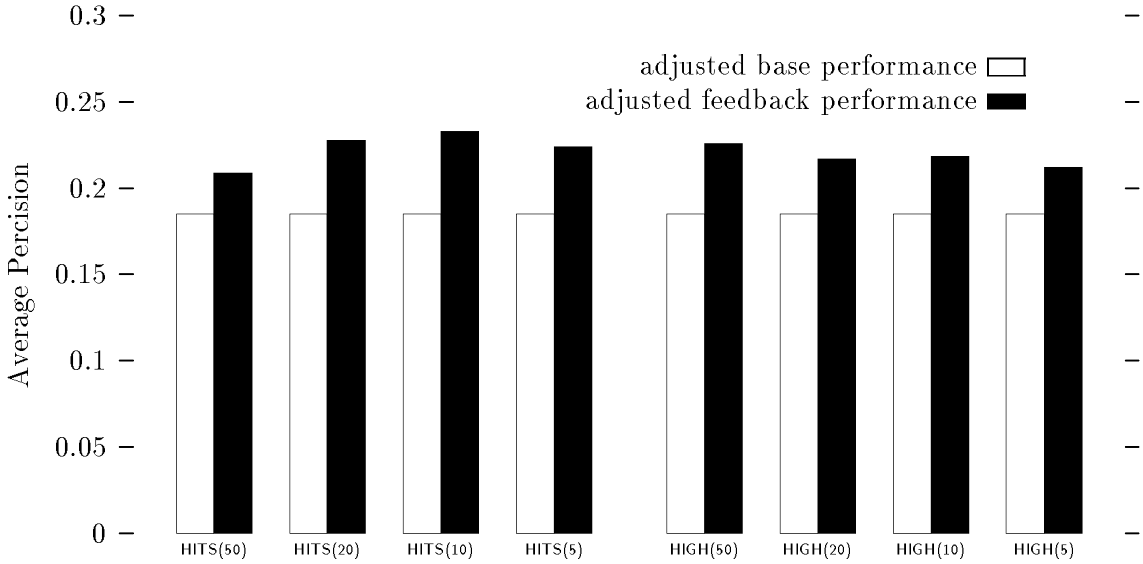


Figure 8: Feedback performance using paragraphs as context units.

as suggested in this paper gives the best performance and is computationally efficient.

Several methods for performing relevance feedback and their performance are described. It is 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. The gain in average precision is as much as 20-25% in some cases. On the other hand, although the data show slight improvement in using terms around hits over high frequency terms, the evidence is not overwhelming.

## Reference


[1] C. Buckley, "Implementation of the SMART Information Retrieval System," *TR 85-686*, Ithaca, N.Y., May 1985.

[2] D. Harman, "Overview of the first Text REtrieval Conference," *Proceedings of the Sixteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, June 1993, 36–47.

[3] Information Dimensions, Inc., "The Complete FIND Handbook," in *The BASISplus Document Set*, Dublin, Ohio, Nov. 1992.

[4] E.M. Keen, "Presenting results of experimental retrieval comparisons," *Information Processing and Management*, Vol. 28, No. 4, 1992, 491–502.

[5] D.L. Lee, "Document Ranking on BASISplus," Information Dimensions, Inc., Final Report, Dublin, Ohio, Sept. 1993, 67 pages.

[6] G. Salton & C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, Vol. 24, No. 5, 1988, 513–523.

[7] G. Salton & C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American Society for Information Science*, Vol. 41, No. 4, 1990, 288–297.

[8] G. Salton & M.J. McGill, *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, N.J., 1983.

[9] Karen Sparck-Jones, ed., *Information Retrieval Experiment.* Butterworth & Co. (Publishers) Ltd., London, 1981.

[10] C. Stanfill & B. Kahle, "Parallel free-text search on the connection machine system," *Communications of ACM*, Vol. 29, No. 12, Dec. 1986, 1229–1239.