Information Processing & Management Vol. 24, No. 1, pp. 5–16, 1988
Printed in Great Britain

0306-4573/88 $3.00 + .00

AN ANALYSIS OF APPROXIMATE VERSUS EXACT DISCRIMINATION VALUES

CAROLYN J. CROUCH*
Computer Science Department, Tulane University, New Orleans, LA 70118, USA

(*Received* 4 *February* 1987; *accepted* 28 *May* 1987)

Abstract—Term discrimination values have been used to characterize and select potential index terms for use during automatic indexing. Two basic approaches to the calculation of discrimination values have been suggested. These approaches differ in their calculation of space density; one method uses the average document-pair similarity for the collection and the other constructs an artificial, "average" document, the centroid, and computes the sum of the similarities of each document with the centroid. The former method has been said to produce "exact" discrimination values and the latter "approximate" values.

This article investigates the differences between the algorithms associated with these two approaches (as well as several modified versions of the algorithms) in terms of their impact on the discrimination value model by determining the differences that exist between the rankings of the exact and approximate discrimination values. The experimental results show that the rankings produced by the exact approach and by a centroid-based algorithm suggested by the author are highly compatible. These results indicate that a previously suggested method involving the calculation of exact discrimination values cannot be recommended in view of the excessive cost associated with such an approach; the approximate (i.e., "exact centroid") approach discussed in this article yields a comparable result at a cost that makes its use feasible for any of the experimental document collections currently in use.

1. INTRODUCTION

1.1 The vector space model

One of the major models in information retrieval is the vector space model. This model views each document in the document collection as a set of unique content terms or word types. Each document is then considered to be a term vector, and the complete document collection becomes a vector space of dimension m, where m is the number of unique terms or word types in the collection. Thus a document vector, $\mathbf{d_j}$, is represented by a set of terms d_{jk}, $1 \leq k \leq m$, where d_{jk} represents the frequency (weight) of term k in document j (i.e., the number of times term k appears in document j). If $d_{jk} = 0$, term k does not appear in document $\mathbf{d_j}$. Queries in the vector space model, like documents, are represented by weighted term vectors.[1]

Given any two term vectors, the similarity between the vectors may be assumed to be inversely related to the angle between them. Thus, if the vectors coincide, the angle between them is zero, and the vectors are identical. In two dimensions, the vector space may be represented by its envelope. The (normalized) document vectors are then viewed as points in the vector space, and the distance between any two points is inversely related to the similarity of the corresponding document vectors. That is, the smaller the distance between two points, the smaller the angle between the corresponding vectors, and the greater the similarity of the vectors in terms of the number of word types they have in common.

Salton et al. [4–6] have shown that the best document space for retrieval purposes is one which maximizes the average separation between documents in the document space.

[1]This paper is based on the conventional view of the vector model, wherein certain assumptions, such as the fact that each term included in a given document or query vector is orthogonal to the other terms, are made [1]. See [2,3] for another view of the vector space model.

*Resources required to support this work were provided by the Computer Science Department of Cornell University.

In such a space, it is easier to distinguish between documents and thus presumably to retrieve documents that are most similar to a given query. The model that allows the terms in a collection to be ranked according to their effect on space density is called the discrimination value model.

1.2 The discrimination value model

The discrimination value model is based on a premise developed by Salton, Yang, and Yu [1]. These researchers felt that a framework that assigned specific roles to single terms, term phrases, and term classes was needed to provide a unified view of content analysis procedures. Such a framework is provided by the discrimination value model, which has the concomitant advantages of offering both a reasonable physical interpretation of the indexing process and a means whereby each potential index term in a collection can be ranked in accordance with its usefulness as a document discriminator.

Consider a collection of documents, each represented by a set of weighted m-dimensional vectors. Then the similarity coefficient computed between any two term vectors can be interpreted as a measure of the closeness or relatedness between the vectors in m-space. If the similarity coefficient is large, the documents are very similar and appear in close proximity to each other in the document space. Contrariwise, if the similarity coefficient is small, the documents exhibit little similarity and are widely separated in the document space.

The discrimination value of a term is then defined as a measure of the change in space separation that occurs when a given term is assigned to the document collection [4]. A "good discriminator" is a term which, when assigned to a document, decreases the space density (i.e., renders the documents less similar to each other). The assignment of a "poor discriminator," on the other hand, increases the density of the space. By computing the density of the document space before and after the assignment of each term, the discrimination value of the term can be determined. The terms can then be ranked in decreasing order of their discrimination values.

Discrimination value has been used by Salton, Yang, and Yu to determine three categories of discriminators, namely, good, poor, and indifferent discriminators. A term with a positive discrimination value has been found to be a good discriminator; Salton et al. suggest that these terms be used directly as index terms. Those terms with negative discrimination values are poor discriminators; the retrieval properties of such terms can be transformed by including them in appropriate phrases. The majority of terms are indifferent disciminators with near-zero discrimination values; the retrieval capabilities of these terms can be enhanced via their incorporation in appropriate thesaurus classes.

If one is concerned with the construction of thesauri or phrase dictionaries, in particular, it is important to know that the terms being considered for inclusion in the thesaurus class or phrase dictionary have the appropriate discrimination values. Theoretically, for example, all terms in a thesaurus class should be indifferent discriminators. Terms used as phrase heads in the phrase construction process should be negative discriminators. And the "best" discriminators, used directly as index terms, are the top n terms when the terms are ranked in decreasing order of their discrimination values. Although the exact placement of each term in the ranking is not critical, the fact that a term lies within a certain range (e.g., the top 20 percent) or has a negative discrimination value can be used to determine the role that term should play in the indexing process.

Two different approaches are used for the calculation of discrimination value. These methods, as well as several modifications that offer substantial operational improvements, are discussed in a following section.

1.3 Similarity measure

A common measure of the similarity between two vectors $\mathbf{d}_j$ and $\mathbf{d}_k$ is the cosine function

$$\cos(\mathbf{d}_j,\mathbf{d}_k) = \frac{\mathbf{d}_j \cdot \mathbf{d}_k}{\|\mathbf{d}_j\| \, \|\mathbf{d}_k\|}$$

where $\mathbf{d_j} \cdot \mathbf{d_k}$ is the inner product and $\|\mathbf{d_j}\|^2 = (\mathbf{d_j} \cdot \mathbf{d_j})$. This relationship may also be expressed as

$$\cos(\mathbf{d_j},\mathbf{d_k}) = \frac{\sum_{i=1}^{m} d_{ji} \cdot d_{ki}}{\sqrt{\sum_{i=1}^{m} d_{ji}^2 \sum_{i=1}^{m} d_{ki}^2}}$$

This is the similarity measure used to compute discrimination value in the context of this article.

2. APPROACHES TO THE CALCULATION OF TERM DISCRIMINATION VALUE

Two approaches to the calculation of discrimination value have been suggested. One approach, the so-called "exact" approach, involves the calculation of all pairwise similarities between the document vectors of the collection. The other, "approximate" approach involves the generation of a centroid vector and the calculation of the similarity of each document with the centroid. A discussion of these two approaches follows.

2.1 The exact approach

A vector space in which the documents are as far apart as possible is a space in which D, the space density, is minimized. Using cosine as the similarity measure, the space density is defined as

$$D = A \sum_{\substack{j=1 \\ j \neq k}}^{n} \sum_{k=1}^{n} \cos(\mathbf{d_j},\mathbf{d_k})$$

D represents the average (pairwise) similarity or density of the document space for some constant A (such as $1/n(n-1)$ [1]). To determine the effect of removing a particular term from the collection, we recompute the density of the document space as follows. Let $\mathbf{d_j^i}$ represent the document vector $\mathbf{d_j}$ with term i removed. Then D_i, the density of the document space with term i removed from the collection, is

$$D_i = A \sum_{\substack{j=1 \\ j \neq k}}^{n} \sum_{k=1}^{n} \cos(\mathbf{d_j^i},\mathbf{d_k^i})$$

and the discrimination value of term i, DV_i, may be calculated for each term i by the equation

$$DV_i = A(D_i - D)$$

2.2 The approximate approach

The second approach to the calculation of discrimination value involves a computation of space density that is based on the sum of the similarities of each document with the centroid of the document space, rather than on the calculation of all pairwise similarities between documents. In this approach, the centroid, $\mathbf{c}$, of the document collection is defined as $\mathbf{c} = (c_1, c_2, \ldots, c_m)$, where

$$c_k = 1/n \sum_{j=1}^{n} d_{jk} \qquad 1 \leq k \leq m$$

The density of the document space, D, can now be calculated using the centroid as

$$D = A \sum_{j=1}^{n} \cos(\mathbf{c},\mathbf{d_j})$$

Let $\mathbf{c}^i$ represent the centroid vector $\mathbf{c}$ with term i removed. The density of the document space with term i removed, D_i, may now be expressed as

$$D_i = A \sum_{j=1}^{n} \cos(\mathbf{c}^i, \mathbf{d}^i_j)$$

and the discrimination value of term i, DV_i, can be computed for each term i again using the formula

$$DV_i = A(D_i - D)$$

Salton [1,5] sees these two approaches as alternative but equivalent approaches to the calculation of term discrimination value, with the centroid approach being more feasible from a computational view. Other authors consider the centroid approach an *approximation* to the exact method (involving the calculation of all pairwise similarities). For example, in a recent article, Willett reports on an algorithm for the calculation of "exact term discrimination values" based on a modification of the basic algorithm that calculates $n^2/2$ pairwise similarities [7].

Although it is obvious that two such different methods, applied to the same collection, will generate *different* term discrimination values, the more pertinent question relates to the differences in the *rankings* of those discrimination values. Following the computation of the discrimination value DV_i for all terms i, the terms can be ranked in decreasing order of discrimination value. Such a ranking will reveal those terms with strongly positive discrimination values (good discriminators that may be used directly as index terms), those terms with near-zero discrimination values (indfferent discriminators whose retrieval performance may be enhanced via their incorporation in thesaurus classes), and the terms with negative discrimination values (poor discriminators, whose retrieval effectiveness may be transformed by their inclusion in appropriate phrases). What differences exist between the discrimination value *rankings* of terms produced via (1) the exact method, involving the computation of all pairwise-similarities, and (2) the approximate or centroid approach?

In this article, the results of an investigation into this question are reported. Four different algorithms to compute discrimination value were implemented. A discussion of these algorithms follows.

3. THE ALGORITHMS

3.1 The basic exact algorithm

The first algorithm for the calculation of term discrimination value involves the computation of $n(n-1)/2$ pairwise document similarities for each of the m word types or unique terms in the collection. This algorithm, which computes exact discrimination values, is herein referred to as the Basic Exact Algorithm. Using a C-like notation, this algorithm may be delineated as shown in Figure 1 (where A represents a constant.)

The complexity of this algorithm is $O(mn^2)$. In experimental document collections, the magnitude of m may well exceed that of n, making this algorithm computationally infeasible for collections of other than trivial size.

3.2 The modified exact algorithm

Regardless of the method used to compute discrimination value, a key factor is the number of documents that actually contain a particular term, say i. In calculating Dv_i in the Basic Exact Algorithm (see Fig. 1), the summation is made over all i, although approximately 50 percent of the index terms will occur in only one document if the terms are distributed according to Zipf's law [8]. In a recent article, Willett [7] suggests a modification of the basic algorithm. using the cosine measure, he defines α, β, and γ as:

If $i \in \mathbf{d}_j, \mathbf{d}_k$, $\alpha = (\mathbf{d}_j \cdot \mathbf{d}_k - \mathbf{d}_j^i \cdot \mathbf{d}_k^i) / \sqrt{(\mathbf{d}_j \cdot \mathbf{d}_j - \mathbf{d}_j^i \cdot \mathbf{d}_j^i) * (\mathbf{d}_k \cdot \mathbf{d}_k - \mathbf{d}_k^i \cdot \mathbf{d}_k^i)} - \cos(\mathbf{d}_j, \mathbf{d}_k)$.

If $i \in \mathbf{d}_k$ only, $\beta = (\mathbf{d}_j \cdot \mathbf{d}_k) / \sqrt{\mathbf{d}_j \cdot \mathbf{d}_j * (\mathbf{d}_k \cdot \mathbf{d}_k - \mathbf{d}_k^i \cdot \mathbf{d}_k^i)} - \cos(\mathbf{d}_j, \mathbf{d}_k)$.

If $i \in \mathbf{d}_j$ only, $\gamma = (\mathbf{d}_j \cdot \mathbf{d}_k) / \sqrt{(\mathbf{d}_j \cdot \mathbf{d}_j - \mathbf{d}_j^i \cdot \mathbf{d}_j^i) * (\mathbf{d}_k \cdot \mathbf{d}_k)} - \cos(\mathbf{d}_j, \mathbf{d}_k)$.

Using these definitions, Willett presents his modification of the Basic Exact Algorithm, which is called the Modified Exact Algorithm (Fig. 2).

```
/* initialize space density D */
D = 0;
/* compute D by calculating n(n - 1)/2
pairwise document similarities */
for (j = 1; j ← n - 1; j++){
    for(k = j + 1; k ← n; k++){
        D = D + cos(dj, dk);
    }
}
/* for each of m word types, compute Dvi, the
space density with term i removed */
for (i = 1; i ← m; i++){
    Dvi = 0;
    for (j=1; j ← n - 1; j++){
        for (k = j + 1; k ← n; k++)
            Dvi = Dvi + cos(dji, dki);
    }
    /* compute DVi, the discrimination value of term i */
    DVi = A * (Dvi - D);
}
```

Fig. 1. The Basic Exact Algorithm

```
/* initialize space density D */
D = 0;
/* initialize m locations to hold Dvi, the space
density with term i removed */
for (i = 1; i ← m; i++){
    Dvi = 0;
}
/* for each of m word types, compute Dvi, the
space density with term i removed */
for (j = 1; j ← n - 1; j++){
    for (k = j + 1; k ← n; k++){
        D = D + cos(dj, dk);
        for (i = 1; i ← m; i++){
            if (dki > 0){
                if (dji > 0)
                    Dvi = Dvi + α; /* i∈dj, i∈dk */
                else
                    Dvi = Dvi + β; /* i ∉ dj, i∈dk */
            }
            else{
                if (dji > 0)
                    Dvi = Dvi + γ; /* i∈dj, i∉dk */
            }
        }
    }
}
/* compute DVi, the discrimination value of term i */
for (i = 1; i ← m; i++){
    DVi = A * Dvi;
}
```

Fig. 2. The Modified Exact Algorithm

In the Modified Exact Algorithm, $n(n-1)/2$ pairwise similarities must be calculated. The complexity of the algorithm is still $O(mn^2)$, but the number of increments (α,β,γ) associated with each similarity calculation depends on the actual number of terms in each vector, which is very small compared to m. Let t_j represent the set of terms contained in $\mathbf{d_j}$ and $|t_j|$ represent the number of elements in the set t_j. Then the number of increments associated with each similarity calculation is $|t_j| + |t_k| - |(t_j \cap t_k)|$, making the total number of increments proportional to τn^2, where τ is the mean number of terms contained in a vector. Moreover, the running time of the algorithm may be substantially reduced by using inverted lists to store the document vectors. However, the algorithm requires m additional storage locations to hold the Dv_i as they are calculated.

3.3 The basic centroid algorithm

In contrast to the exact algorithms presented in Figures 1 and 2, the other common approach to the calculation of discrimination value is based on calculating the pairwise similarity of each document in the collection with the centroid vector. This Basic Centroid Algorithm, of complexity $O(mn)$, is presented in Figure 3.

3.4 The modified centroid algorithm

Crawford [9], having noted that few documents actually contain term i and using cosine as his similarity measure, suggests a revision of the Basic Centroid Algorithm, called the Modified Centroid Algorithm (Fig. 4).

The Modified Exact Algorithm is based on three modifications: (1) The summation of Dv_i in Figure 3 is made over all documents. Because few documents contain term i, $\cos(\mathbf{c}^i,\mathbf{d}_j^i)$ is identical to $\cos(\mathbf{c},\mathbf{d_j})$ in most cases. To compute Dv_i, for each document j in which term i occurs (i.e., $d_{ij} \neq 0$), Crawford computes Dv_i using the formula

$$Dv_i = D - \cos(\mathbf{c},\mathbf{d_j}) + \cos(\mathbf{c}^i,\mathbf{d}_j^i)$$

Although the purpose of this modification is to reduce the number of similarity coefficients that must be calculated to compute Dv_i, this modification insures that the discrimination values calculated via this algorithm will differ somewhat from those produced by the centroid algorithm of Figure 3. The discrepancy comes from the fact that for each

```
/* calculate centroid vector c
with coefficients c_k */
for (k = 1;k ← m;k++){
    c_k = 0;
    for (j = 1;j ← n;j++){
        c_k = c_k + d_jk;
    }
    c_k = c_k/n;
}
/* initialize space density D */
D = 0;
/* compute D as the sum of the similarities
of each document vector d with c */
for (j = 1;j ← n;j++){
    D = D + cos(c,d_j);
}
/* for each of m word types, compute Dv_i, the
space density with term i removed */
for (i = 1;i ← m;i++){
    Dv_i = 0;
    for (j = 1;j ← n;j++){
        Dv_i = Dv_i + cos(c^i,d_j^i);
    }
    /* compute DV_i, the discrimination value of term i */
    DV_i = A * (Dv_i - D);
}
```

Fig. 3. The Basic Centroid Algorithm

```
/* calculate centroid vector c
with coefficients c_k */
for (k = 1; k ← m; k++) {
    c_k = 0;
    for (j = 1; j ← n; j++) {
        c_k = c_k + d_jk;
    }
    c_k = c_k/n;
}
/* initialize space density D */
D = 0;
/* compute D as the sum of the similarities
of each document vector d with c */
for (j = 1; j ← n; j++) {
    D = D + cos(c, d_j);
}
/* for each of m word types, compute Dv_i, the
space density with term i removed */
for (i = 1; i ← m; i++) {
    /* initialize Dv_i to D */
    Dv_i = D;
    for (j = 1; j ← n; j++) {
        if (d_ij > 0) /* if i ∈ d_j */
            Dv_i = Dv_i − cos(c, d_j) + cos(c^i, d_j^i);
    }
    /* compute DV_i, the discrimination value of term i */
    DV_i = A * (Dv_i − D);
}
```

Fig. 4. The Modified Centroid Algorithm

document $\mathbf{d_j}$, if term i is not found in $\mathbf{d_j}$, then the contribution of term i to Dv_i is computed as

$$\cos(\mathbf{c},\mathbf{d_j}) = \mathbf{c}\cdot\mathbf{d_j}/\sqrt{\mathbf{c}\cdot\mathbf{c} * \mathbf{d_j}\cdot\mathbf{d_j}}$$

rather than its actual value of

$$\cos(\mathbf{c},\mathbf{d_j}) = \mathbf{c}\cdot\mathbf{d_j}/\sqrt{(\mathbf{c}\cdot\mathbf{c} - \mathbf{c^i}\cdot\mathbf{c^i}) * \mathbf{d_j}\cdot\mathbf{d_j}}$$

(2) Crawford suggests the storing of partial sums calculated during the execution of $D = D + \cos(\mathbf{c},\mathbf{d_j})$; that is,

$$\cos(\mathbf{c},\mathbf{d_j}) = \mathbf{c}\cdot\mathbf{d_j}/\sqrt{\mathbf{c}\cdot\mathbf{c} * \mathbf{d_j}\cdot\mathbf{d_j}}$$

and the values of $\mathbf{c}\cdot\mathbf{d_j}$ and $\mathbf{d_j}\cdot\mathbf{d_j}$ are stored for each j, $j = 1, \ldots, n$, as the space density D is calculated. A buffer of $2n$ locations is required to hold these values, which are then used rather than recomputed during the calculation of Dv_i. (Note that $\mathbf{c}\cdot\mathbf{c}$ is a function of the centroid and is hence a constant that need be calculated only once.)

(3) The third refinement suggested by Crawford is the use of inverted lists to hold the document vectors.

Although the complexity of Crawford's algorithm is still $O(mn)$, the actual number of computations involved in calculating the similarity coefficients is greatly reduced, since a new Dv_i is computed only if term i is actually present in document j and the storage of partial sums makes much of the arithmetic unnecessary. The use of an inverted file further reduces the execution time.

3.5 The exact centroid algorithm

Another possibility now presents itself. Consider using the centroid method and storing the partial sums during the calculation of the space density as suggested by Crawford. Define α and β as follows:

$$\text{if } i \in \mathbf{d}_j, \qquad \alpha = (\mathbf{c}\cdot\mathbf{d}_j - \mathbf{c}^i\cdot\mathbf{d}_j^i)/\sqrt{(\mathbf{c}\cdot\mathbf{c} - \mathbf{c}^i\cdot\mathbf{c}^i) * (\mathbf{d}_j\cdot\mathbf{d}_j - \mathbf{d}_j^i\cdot\mathbf{d}_j^i)}.$$

$$\text{if } i \notin \mathbf{d}_j, \qquad \beta = \mathbf{c}\cdot\mathbf{d}_j/\sqrt{(\mathbf{c}\cdot\mathbf{c} - \mathbf{c}^i\cdot\mathbf{c}^i) * (\mathbf{d}_j\cdot\mathbf{d}_j)}$$

where $\mathbf{c}\cdot\mathbf{d}_j$ and $\mathbf{d}_j\cdot\mathbf{d}_j$ for $j = 1, \ldots, n$ are the values saved (at a cost of $2n$ storage locations) during calculation of the centroid. (Recall that $\mathbf{c}\cdot\mathbf{c}$ is computed only once.) We now define the Exact Centroid Algorithm (see Fig. 5). This algorithm represents a modification of the Basic Centroid Algorithm which allows the "exact" space density associated with the removal of term i to be calculated, rather than an approximation to it (as in the case of the Modified Centroid Algorithm).

This approach increases the number of similarity coefficients that must be computed during the calculation of the Dv_i over those produced by the Modified Centroid Algorithm, since Dv_i is modified by either α or β (depending on whether term i is or is not contained in $\mathbf{d}_j$), rather than simply requiring the calculation of two similarity coefficients for each term i contained in $\mathbf{d}_j$. Hence, the execution time of this algorithm will be somewhat longer than that of the Modified Centroid Algorithm (assuming that both algorithms use inverted lists), and the ranks of the resulting discrimination values can be expected to differ from those produced by that algorithm.

To determine the differences in (1) ranks of the resulting discrimination values and (2) execution times of the various algorithms, the algorithms were run on three different document collections. The result of these investigations are presented as follows.

4. THE EXPERIMENTS

4.1 Methodology

The following algorithms were implemented:

1. The Basic Exact Algorithm (Fig. 1).
2. The Modified Exact Algorithm (Fig. 2).
3. The Modified Centroid Algorithm (Fig. 4).
4. The Exact Centroid Algorithm (Fig. 5).

```
/* calculate centroid vector c
with coefficients c_k */
for (k = 1; k ← m; k++) {
    c_k = 0;
    for (j = 1; j ← n; j++) {
        c_k = c_k + d_jk;
    }
    c_k = c_k/n;
}
/* initialize space density D */
D = 0;
/* compute D as the sum of the similarities
of each document vector d with c */
for (j = 1; j ← n; j++) {
    D = D + cos(c, d_j);
}
/* for each of m word types, compute Dv_i, the
space density with term i removed */
for (i = 1; i ← m; i++) {
    Dv_i = 0;
    for (j = 1; j ← n; j++) {
        if (d_ij > 0)
            Dv_i = Dv_i + α; /* i ∈ d_j */
        else
            Dv_i = Dv_i + β; /* i ∉ d_j */
    }
    /* compute DV_i, the discrimination value of term i */
    DV_i = A * (Dv_i − D);
}
```

Fig. 5. The Exact Centroid Algorithm

The basic algorithm was run, for purposes of comparison, only on the smallest document collection (ADI). The time required to process even this very small collection clearly indicates that this approach is prohibitive for collections of larger size. Each of the three remaining algorithms was run on each of the three test collections: ADI, Medlars, and CACM. The characteristics of these collections are found in Table 1. (For more complete descriptions of these collections, see [10,11,12].) The collections are available through the Smart experimental retrieval system at Cornell University.

4.2 Empirical results

The amount of time taken to execute each algorithm on each collection is specified in Table 2. Inverted lists were used in each algorithm to improve the execution times. The same similarity measure (cosine) was used in each case.

Table 2 demonstrates that the Modified Exact Algorithm is indeed a significant improvement over the Basic Exact Algorithm. It also shows that the Modified Exact Algorithm, of order $O(mn^2)$, is still extremely expensive compared with the centroid approaches (both $O(mn)$). This is due to the fact that, although the execution time is reduced by the modifications as previously described, the cost associated with the $n(n-1)/2$ similarity coefficients that must still be calculated cannot be reduced and dominates the algorithm. In the centroid approaches, the n similarity coefficients (of $\mathbf{c}$ with $\mathbf{d}_j$, $j = 1, \ldots, n$) that are computed during the calculation of the space density D can be "reused" later in the algorithm when saved as partial sums. The cost associated with the calculation of these similarity coefficients becomes even greater if inverted lists, which can substantially decrease processing costs when used with a suitable similarity measure (such as cosine), can no longer be used. Some similarity measures (e.g., $p$norm) preclude the use of inverted lists. The cost associated with the computation of the similarity coefficients is the major component reflected in the figures of Table 2.

Given that the costs associated with the Modified Exact Algorithm are overwhelmingly greater than those of the centroid algorithms, the only reason for choosing the former method over the latter would be the supposed superiority of the resulting discrimination values (i.e., the "exact" versus the "approximate"). To determine the differences in the ranks of the discrimination values produced by each of the algorithms, the resulting discrimination values were sorted in descending order, and the Spearman rank correlation [13] was computed between the values produced by the base case (the Modified Exact Algorithm) and those produced by the Modified Centroid and the Exact Centroid Algorithms. The results are shown in Table 3.

Table 1. Collection statistics

Collection	Number of vectors (n)	Number of terms (m)	Mean number of terms per vector
ADI	82	822	25.5
Medlars	1033	6927	51.6
CACM	3204	8503	27.5

Table 2. Time statistics*

Algorithm	Collections		
	ADI	Medlars	CACM
Basic Exact (Fig. 1)	7,385	—	—
Modified Exact (Fig. 2)	219	121,640	25,6148
Modified Centroid (Fig. 4)	14	2,468	8,104
Exact Centroid (Fig. 5)	38	4,379	14,765

*CPU time in seconds.

Table 3. Spearman rank correlations

Algorithm	Collections		
	ADI	Medlars	CACM
Basic Exact (Fig. 1)	1.00	—	—
Modified Exact (Fig. 2)	1.00	1.00	1.00
Modified Centroid (Fig. 4)	0.66	0.84	0.91
Exact Centroid (Fig. 5)	1.00	1.00	0.98

Table 3 indicates that the rankings of the discrimination values produced by the Modified Exact and the Exact Centroid Algorithms are very similar, with the ranking produced by the Modified Centroid Algorithm being somewhat less similar to the base case. The particular terms of interest are those occurring in the top n percent of the collection (which are used without transformation as index terms) and those with negative discrimination values (which may be transformed into more desirable index terms via their incorporation in phrases). Once these two sets are identified, the remaining terms (indifferent discriminators) are candidates for inclusion in thesaurus classes. To determine the actual similarity between the rankings, the top n percent of the terms produced by each centroid algorithm were compared to the corresponding top n percent of the terms produced by the Modified Exact Algorithm. The results of these comparisons are found in Table 4.

When the terms produced by the centroid algorithms are ranked in decreasing order of discrimination value and the top n percent of the terms in each set are examined, the Exact Centroid Algorithm produces a set of terms that has more terms in common with the top n percent of the terms in the base case than does the set produced by the Modified Centroid Algorithm. This is true for all values of n and over all three collections (ADI, Medlars, and CACM). Moreover, a statistical test shows that the difference between the proportions of terms matching with the base case is highly significant.

Consider now the sets of negative discriminators produced by each algorithm. When these sets are examined, one finds that whereas the sets of negative discriminators produced by the Exact Centroid Algorithm correspond very closely to the sets produced by the Modified Exact Algorithm for all three collections, the sets produced by the Modified Centroid Algorithm are much larger in each case. Table 5 shows the number of negative discriminators produced by each algorithm, as well as the number of terms in each set which are identical to terms in the base case.

Table 5 shows that for each collection, the set of negative discriminators generated by the Exact Centroid Algorithm is very similar in size to the corresponding set produced by the Modified Exact Algorithm. Moreover, the sets themselves exhibit a high degree of overlap. The sets of negative discriminators produced by the Modified Centroid Algorithm, on the other hand, are much larger than the corresponding sets produced by the Modified Exact algorithm (the base case). The true negative discriminators (those produced by the base case) are subsets of these sets, but were one to identify poor discriminators on the basis of negative discrimination value, very different sets would be produced by the Modified

Table 4. Comparison of terms in the top n percent*

Algorithm	n Percent		
	10	20	30
Modified Centroid (Fig. 4)	71	77	82
Exact Centroid (Fig. 5)	95	96	96

*All comparisons are made to corresponding n percent of the terms produced by the Modified Exact Algorithm (Fig. 2) and are averaged over all three collections.

Table 5. Negative discriminators produced by each algorithm*

Algorithm	Collections		
	ADI	Medlars	CACM
Modified (Exact (Fig. 2)	32	157	33
Modified Centroid (Fig. 4)	122 (32*)	398 (157*)	147 (33*)
Exact Centroid (Fig. 5)	33 (32*)	158 (155*)	31 (25*)

*Number of terms in common with corresponding sets produced by Modified Exact Algorithm (Fig. 2).

Centroid Algorithm as opposed to the Modified Exact or Exact Centroid Algorithms. An examination of the sets of negative discriminators produced by the Modified Centroid Algorithm reveals that the large numbers of terms that become negative discriminators under this algorithm are essentially medium frequency terms (i.e., that these terms are good discriminators in the sets produced by the Modified Exact and Exact Centroid Algorithms).

Given (1) the cost differential between the Modified Exact and the centroid algorithms and (2) the very similar rankings produced by the Modified Exact and the Exact Centroid algorithms, there appears to be no reason to consider a method involving the calculation of n^2 pairwise similarities (e.g., the Modified Exact Algorithm) when calculating discrimination value. Both the Modified Centroid and the Exact Centroid methods are much less expensive, but the rankings of terms produced by the former method differ significantly from those of the base case. Moreover, since the Modified Centroid Algorithm produces a distorted picture of the document space in terms of the number of medium frequency terms that are moved into the ranks of poor discriminators, this method, although somewhat less expensive than the Exact Centroid Algorithm, cannot be recommended. The experimental evidence indicates that the Modified Exact Algorithm and the Exact Centroid Algorithm in fact yield equivalent results and that the latter is overwhelmingly more efficient.

5. CONCLUSION

Two algorithms for the calculation of term discrimination value, the Basic Exact Algorithm (Fig. 1) and the Basic Centroid Algorithm (Fig. 3), as well as several modifications to these algorithms, are investigated. Three feasible algorithms, the Modified Exact Algorithm (Fig. 2—the base case), the Modified Centroid Algorithm (Fig. 4), and the Exact Centroid Algorithm (Fig. 5), are examined in detail. The Modified Centroid Algorithm is found to produce a ranking of terms that varies significantly from that produced by the base case. It also produces a distorted view of the document space by moving some medium frequency terms into the ranks of negative discriminators. On the other hand, the results produced by the Exact Centroid Algorithm are highly compatible with those produced by the Modified Exact Algorithm. The experimental results indicate that the rankings of the terms produced by the Exact Centroid Algorithm are so similar to those produced by the Modified Exact Algorithm and the differences in execution time are of such a magnitude as to render the use of any current non-centroid approach to the computation of discrimination value infeasible.

Acknowledgments—The author is indebted to the members of the Smart group at Cornell University for their suggestions and constructive criticism. My thanks go to Professors Gerard Salton and Donald Crouch and to Chris Buckley, Maria Smith, and Joel Fagin. I am especially indebted to Ellen Voorhees, whose aid was invaluable in my initial dealings with the Smart system.

REFERENCES

1. Salton, G.; McGill, M. Introduction to modern information retrieval. New York: McGraw-Hill; 1983.
2. Wong, S.K.M.; Raghavan, V. Vector space model of information retrieval—a reevaluation," Proceedings