On the Complexity of Rocchio's Similarity-Based Relevance Feedback Algorithm

Zhixiang Chen and Bin Fu
Department of Computer Science, University of Texas-Pan American, 1201 W. University Drive, Edinburg, TX 78541-2999. E-mail: {chen, binfu}@cs.panam.edu

Rocchio's similarity-based relevance feedback algorithm, one of the most important query reformation methods in information retrieval, is essentially an adaptive learning algorithm from examples in searching for documents represented by a linear classifier. Despite its popularity in various applications, there is little rigorous analysis of its learning complexity in literature. In this article, the authors prove for the first time that the learning complexity of Rocchio's algorithm is $O(d + d^2(\log d + \log n))$ over the discretized vector space $\{0, \ldots, n-1\}^d$, when the inner product similarity measure is used. The upper bound on the learning complexity for searching for documents represented by a monotone linear classifier $(\vec{q}, 0)$ over $\{0, \ldots, n-1\}^d$ can be improved to, at most, $1 + 2k(n-1)(\log d + \log(n-1))$, where k is the number of nonzero components in $\vec{q}$. Several lower bounds on the learning complexity are also obtained for Rocchio's algorithm. For example, the authors prove that Rocchio's algorithm has a lower bound $\Omega(\binom{d}{2}\log n)$ on its learning complexity over the Boolean vector space $\{0, 1\}^d$.

Introduction

Information retrieval has a long history of research on relevance feedback (for example, Baeza-Yates & Ribeiro-Neto, 1999; Frake & Baeza-Yates, 1992; Ide, 1971a, 1971b; Raghavan & Wong, 1986; Rocchio, 1971; Salton, 1989; van Vijsbergen, 1979), and becomes a necessary part of our daily life due to the popularity of the Web. It is regarded as the most popular query reformation strategy (Baeza-Yates & Ribeiro-Neto, 1999; Salton, 89). The central idea of relevance feedback is to improve search performance for a particular query by modifying the query systematically, based on the user's judgments of the relevance or irrelevance of some of the documents retrieved. In the vector space model (Salton, 1989; Salton, Wong, & Yang, 1975), both documents and queries are represented as vectors in a discretized vector space. In this case, relevance feedback is essentially an adaptive learning algorithm from examples (Chen & Zhu, 2002): A query vector and a similarity measure are used to classify documents as relevant and irrelevant; the user's judgments of the relevance or irrelevance of some the classified documents are used as examples for updating the query vector as a linear combination of the initial query vector and the examples judged by the user.

In his popular textbook, van Vijsbergen (1979) describes the relevance feedback as a fixed error correction procedure and relates it to the linear classification problem. When the inner product similarity is used, relevance feedback is just a "perceptron-like" learning algorithm (Lewis, 1991). There is an optimal way for updating the query vector if the sets of relevant and irrelevant documents are known (Rocchio, 1971). Practically, it is impossible to derive the optimal query vector because the full sets of the relevant and irrelevant documents are not available. Wong, Yao, and Bollmann (1988) studied the linear structure of user preference in information retrieval. They designed a very good gradient descent procedure to compute the coefficients of a linear function and analyzed its performance. To update the query vector adaptively, their gradient descent procedure must know the user preference, which is, in practice, the unknown target to be learned by an information retrieval system. Chen (2001, 2004), devised multiplicative adaptive algorithms for user-preference retrieval with provable, efficient performance.

There are many different variants of relevance feedback in information retrieval. However, in this article we only study Rocchio's similarity-based relevance feedback algorithm (Rocchio, 1971; Salton, 1989). Despite its popularity in various applications, there is little rigorous formal analysis of its complexity as a learning algorithm in literature. As a first step towards formal analysis of Rocchio's similarity-based relevance feedback algorithm, the work in (Chen & Zhu, 2002) establishes a linear lower bound on the learning complexity for the algorithm in searching for documents represented by a monotone linear classifier (which is equivalently a disjunction of k relevant features) $x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k}$ over the Boolean vector space $\{0, 1\}^d$, when any of the four typical

Received September 12, 2005; revised October 16, 2006; accepted October 16, 2006

© 2007 Wiley Periodicals, Inc. • Published online 31 May 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.20612

similarity measures (inner product, dice coefficient, cosine coefficient, and Jaccard coefficient) listed in Salton (1989) is used. The linear lower bound obtained in (Chen & Zhu, 2002) is independent of the query updating factors and the classification threshold that are used by the algorithm. A number of challenging problems regarding further analysis of the algorithm remain open (Chen & Zhu, 2002). In practice, a fixed query-updating factor and a fixed classification threshold are often used in Rocchio's similarity-based relevance feedback algorithm (Baeza-Yates & Ribeiro-Neto, 1999; Salton, 1989).

Using a fixed query-updating factor has many merits, such as simplicity and efficiency. As another example, the popular Winnow algorithm (Littlestone, 1988) uses a fixed updating factor. When this is the case, the work in Chen and Fu (2005) strengthens the linear lower bound in (Chen & Zhu, 2002) to a quadratic lower bound for Rocchio's algorithm in searching for documents represented by $x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k}$ over the Boolean vector space $\{0, 1\}^d$.

In this article, we prove for the first time that the learning complexity of Rocchio's algorithm is $O(d + d^2(\log d + \log n))$ over the discretized vector space $\{0, \ldots, n-1\}^d$, when the inner product similarity measure is used. The upper bound on the learning complexity for searching for documents represented by a monotone linear classifier $(\vec{q}, 0)$ over $\{0, \ldots, n-1\}^d$ can be improved to at most $1 + 2k(n-1)(\log d + \log(n-1))$, where k is the number of nonzero components in $\vec{q}$. An $\Omega(\binom{d}{2}\log n)$ lower bound on the learning complexity is obtained for Rocchio's algorithm over the vector space $\{0, \ldots, n-1\}^d$. In practice, Rocchio's algorithm often uses fixed query updating factors. When this is the case, the above lower bound is strengthened to $2^{\Omega(d)}$ over the Boolean vector space $\{0, 1\}^d$. In general, if the query updating factors are bounded by $O(n^c)$, for some constant $c \geq 0$, an $\Omega(n^{d-1-c}/(n-1))$ lower bound is obtained over $\{0, \ldots, n-1\}^d$. The rest of the article is organized as follows. In the next section, we give a formal presentation of Rocchio's similarity-based relevance feedback algorithm. In the third section, we prove a general upper bound on the learning complexity for Rocchio's algorithm. Then, we show that the general upper bound obtained in the third section can be improved for Rocchio's algorithm in the case of searching for documents represented by a monotone linear classifier over $\{0, \ldots, n-1\}^d$. We go on to prove several lower bounds on the learning complexity of Rocchio's algorithm. We present our conclusions in the final section.

Rocchio's Similarity-Based Relevance Feedback Algorithm

Let R be the set of all real values, and R^+ be the set of all nonnegative real values. Let d and n be two integers with $d \geq 1$ and $n \geq 2$. In the Boolean vector space model in information retrieval (Salton, 1989; Salton, Wong, & Yang, 1975), a collection of d features (or terms) $T_1, T_2, \ldots, T_d$ are used to represent documents and queries. Each document $\boldsymbol{x}$ is represented as a vector $\vec{v}_x = (x_1, x_2, \ldots, x_d)$ such that for any i, $1 \leq i \leq d$, the ith component of $\vec{v}_x$ is one if the ith feature T_i appears in $\boldsymbol{x}$ or zero otherwise. Each query $\boldsymbol{q}$ is represented by a vector $\vec{v}_q = (q_1, q_2, \ldots, q_d)$ such that for any i, $1 \leq i \leq d$, the ith component of $\vec{v}_q$ is a real value used to determine the relevance (or weight) of the ith feature T_i. Because of the unique vector representations of documents and queries, for convenience we simply use $\vec{x}$ and $\vec{q}$ to stand for their vector representations $\vec{v}_x$ and $\vec{v}_q$, respectively. If term frequencies are used to index a document $\boldsymbol{x}$, then $\vec{x}$ is a vector in the discretized vector space $\{0, \ldots, n-1\}^d$. Note that fractional term frequency/inverse document frequency vectors can converted into vectors in $\{0, \ldots, n-1\}^d$.

A similarity measure, called *similarity* for short, in general is a function m from $R^d \times R^d$ to R^+. A similarity m is used to determine the *relevance closeness* of documents to the search query and to rank the documents according to such closeness. In the Boolean vector space model of information retrieval (Baeza-Yates & Ribeiro-Neto, 1999; Salton, 1989; Salton, Wong, & Yang, 1975), to retrieve relevant documents for a given query vector $\vec{q}$ with respect to a similarity m, the system searches for all the documents $\vec{x}$, classifies those with similarity values $m(\vec{q}, \vec{x})$ higher than an explicit or implicit threshold as relevant, and returns to the user a short list of relevant documents with highest similarity values. This information retrieval process is, in fact, determined by a classifier, which is composed of a query vector $\vec{q}$, a similarity m, and a real-valued threshold ϕ. Among a variety of similarity measures, vector inner product similarity is commonly used (Baeza-Yates & Ribeiro-Neto, 1999; Salton, 1989; Salton, Wong, & Yang, 1975). To simplify presentation, we will focus on vector inner product similarity, $\vec{q} \cdot \vec{x} = q_1x_1 + q_2x_2 + \cdots + q_dx_d$, throughout this article.

Unfortunately, in the real world of information retrieval applications, usually an ideal query vector cannot be generated due to many factors, such as the limited knowledge of the users about the whole document collection. A typical example is the real-world problem of Web search. In such a case, the user may use a few keywords to express what documents are wanted. However, it is nontrivial for the user and a Web search engine to define *precisely* the collection of documents wanted as a query vector composed of a set of keywords. The alternative solution to the query formation problem is, as stated in Salton (1989), to conduct searches iteratively, first operating with a tentative query formation (i.e., an initial query vector), and then improving formations for subsequent searches based on evaluations of the previously retrieved materials. This type of method for automatically generating improved query formation is called *relevance feedback*, and one particular and well-known example is Rocchio's similarity-based relevance feedback (Rocchio, 1971; Ide, 1971b; Salton, 1989).

Rocchio's similarity-based relevance feedback algorithm works in a step-by-step adaptive refinement fashion as follows. Starting at an initial query vector $\vec{q}_1$, the algorithm searches for all the documents $\vec{x}$ such that $\vec{x}$ is *very close* to $\vec{q}_1$ according to the similarity $\vec{q}_1 \cdot \vec{x}$, ranks them by the similarity, and finally presents a short list of the top-ranked documents to the user. The user examines the returned list of documents and judges some of the documents as relevant or irrelevant. At step $t \geq 1$, assume that the list of documents the user judged is $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$. Then, the algorithm updates its query vector as $\vec{q}_t = \alpha_{t_0}\vec{q}_1 + \sum_{j=1}^{t-1}\alpha_{t_j}\vec{x}_j$, where the coefficients $\alpha_{t_j} \in \mathrm{R}$ for $j = 0, 1, \ldots, t-1$. The algorithm then uses the updated query vector $\vec{q}_t$ to search for relevant documents, ranks the documents according to their similarity to $\vec{q}_t$, and presents the top-ranked documents to the user. In practice, a threshold ϕ is explicitly (or implicitly) used to select the highly ranked documents. Practically, the coefficients α_{t_j} may be fixed as 1, -1, or 0.5 (Baeza-Yates & Ribeiro-Neto, 1999; Salton, 1989).

The similarity-based relevance feedback algorithm is essentially an adaptive learning algorithm from examples (Chen & Fu, 2005; Chen & Zhu, 2002; Lewis, 1991; Salton & Buckley, 1990; van Vijsbergen, 1979). The goal of the algorithm is to learn some unknown classifier to classify documents as relevant or irrelevant. The learning is performed by modifying (or updating) the query vector that serves as the hypothetical representation of the collection of all relevant documents. The method for updating the query vector is similar to the Perceptron algorithm for multivalued features (Hampson & Volper, 1990; Rosenblatt, 1958), where updating factors are allowed to be 1 or -1, while arbitrary updating factors are, in general, permitted for relevance feedback. We give the necessary formal definitions in the following.

Definition 1: A linear classifier over the d-dimensional discretized vector space $\{0, \ldots, n-1\}^d$ is a pair $(\vec{q}, \phi)$, where $\vec{q} \in \mathrm{R}^d$ is the query/weight vector and $\phi \in \mathrm{R}$ is a classification threshold. The classifier classifies any documents $\vec{x} \in \{0, 1, \ldots, n-1\}^d$ as relevant if $\vec{q} \cdot \vec{x} \geq \phi$ or irrelevant otherwise.

Definition 2: An adaptive learning algorithm A for learning an unknown target linear classifier $(\vec{q}, \phi)$ over the d-dimensional discretized vector space $\{0, \ldots, n-1\}^d$ from examples is a game played between the algorithm A and the user in a step-by-step fashion, where the query/weight vector $\vec{q}$ and the threshold ϕ are unknown to the algorithm A. At any step $t \geq 1$, A gives a linear classifier $(\vec{q}_t, \phi_t)$ as a hypothesis to the target linear classifier to the user, where $\vec{q}_t \in \mathrm{R}^d$ and $\phi_t \in \mathrm{R}$. If the hypothesis is equivalent to the target, then the user says "yes" to conclude the learning process. Otherwise, the user presents a counterexample $\vec{x}_t \in \{0, \ldots, n-1\}^d$ such that the target classifier and the hypothetical classifier differ at $\vec{x}_t$. In this case, we say that the algorithm A makes a mistake. At step $t+1$, the algorithm A constructs a new hypothetical linear classifier $(\vec{q}_{t+1}, \phi_{t+1})$ to the user based on the received counterexamples $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_t$. The learning complexity (or the mistake bound) of the algorithm A is in the worst case the maximum number of counterexamples that it may receive from the user to learn a classifier over $\{0, \ldots, n-1\}^d$.

If the readers are familiar with online learning from equivalence queries, for example, (Littlestone, 1988; Maass & Turán, 1994), then an adaptive learning algorithm as defined above is a proper online learning algorithm for learning the class of classifiers from equivalence queries over the d-dimensional Boolean or discretized vector space.

Following Chen and Zhu (2002), we now give the formal definition of Rocchio's similarity-based relevance feedback algorithm.

Definition 3: Rocchio's similarity-based relevance feedback algorithm is an adaptive learning algorithm for learning any linear classifier $(\vec{q}, \phi)$ over the d-dimensional discretized vector space $\{0, \ldots, n-1\}^d$ from examples, where both the query vector $\vec{q}$ and the threshold ϕ are unknown to the algorithm, but the inner product similarity measure is known. Let $(\vec{q}_0, \phi_1)$ be the initial hypothesis. Assume that at the beginning of step $t > 1$ the algorithm has received a sequence of counterexamples $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$, then the algorithm uses the following modified query vector $\vec{q}_t$ for its next classification:

$$\vec{q}_t = \alpha_{t_0}\vec{q}_1 + \sum_{j=1}^{t-1}\alpha_{t_j}\vec{x}_j, \tag{1}$$

where $\alpha_{t_j} \in R$, for $j = 0, 1, \ldots, t-1$, are called *query updating factors*.

Note that our definition above is a generalized version of Rocchio's original algorithm. In our definition, arbitrary real values can be used as query updating factors in computing the updated query vector; and our definition allows adaptive learning until the target is obtained. Also note that the perceptron algorithm for multivalued features (Hampson & Volper, 1990) allows updating factors to be 1 or -1.

An Upper Bound for Documents Represented by a Linear Classifier

In this section, we will prove an upper bound on the learning complexity of Rocchio's similarity-based algorithm in searching for documents represented by a linear classifier over the discretized vector space $\{0, \ldots, n-1\}^d$. We first prove Lemma 1 using linear independence to allow Rocchio's algorithm to simulate any adaptive learning algorithm for learning linear classifiers. Utilizing linear independence to derive upper bounds on learning complexity can be found, for example, in Bshouty, Chen, Decatur, and Homer (1995), Chen and Homer (1997), Kivinen, Warmuth, and Auer (1997).

Lemma 1: Let A be any given adaptive learning algorithm for learning linear classifiers over the d-dimensional discretized vector space $\{0, \ldots, n-1\}^d$. Assume that A issues linear classifiers as its hypotheses. Let $l(A)$ and $t(A)$ denote, respectively, the learning complexity and the time complexity of the algorithm A. Then, the learning complexity of Rocchio's similarity-based relevance feedback in searching for documents represented by a linear classifier over $\{0, \ldots, n-1\}^d$ is at most $d + l(A)$, and its time complexity is $O(d^2 \log^2 n\ (d + l(A)) + t(A))$.

Proof: By Definition 2, the algorithm A works in a step-by-step fashion. At step $t \geq 1$, A computes a hypothesis linear classifier $(\vec{q}_t, \phi_t)$. We design the following procedure to allow Rocchio's algorithm to simulate the algorithm A:

Simulation Procedure:

```
For $t = 1$, do
        Call $A$ to generate the hypothesis $(\vec{q}_1, \phi_1)$.
        Set $\vec{q}_1{}^* = \vec{0}$ and let Rocchio's algorithm present
            the hypothesis $(\vec{q}_1{}^*, \phi_1)$ to the user.
        If the user answers yes, then stop. Otherwise, a
            counterexample $\vec{x}_1$ is received as relevance
            feedback judged by the user.
For $t > 1$, do
        If $\vec{x}_{t-1}$ is also a counterexample to $A$'s current
            hypothesis linear classifier $(\vec{q}_{t-1}, \phi_{t-1})$, then
            call $A$ to generate a new hypothesis $(\vec{q}_t, \phi_t)$
            using $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$.
        If $\vec{x}_{t-1}$ is not a counterexample to $A$'s current
            hypothesis linear classifier $(\vec{q}_{t-1}, \phi_{t-1})$, then
            simply let $\vec{q}_t = \vec{q}_{t-1}$ and $\phi_t = \phi_{t-1}$.
        Compute $\vec{q}_t{}^*$ as the projection of $\vec{q}_t$ onto the linear
            space defined by $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$.
        Let Rocchio's algorithm present the new hypoth-
            esis $(\vec{q}_t{}^*, \phi_t)$ to the user.
        If the user answers yes, then stop. Otherwise a
            counterexample $\vec{x}_t$ is received as relevance
            feedback, then repeat the process for $t > 1$.
```

We note that Rocchio's algorithm in the above simulation procedure uses a zero initial query vector. For any $t > 1$, the algorithm uses a query vector $\vec{q}_t{}^* = \sum_{i=1}^{t-1} \alpha_i \vec{x}_i$ for some $\alpha_i \in R$, because $\vec{q}_t{}^*$ is the projection of $\vec{q}_t$ onto the linear space defined by $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$. This means that the query vector is updated following Expression (1). Also, since $\vec{q}_t{}^*$ is the projection of $\vec{q}_t$ onto the linear space defined by $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$, we have $\vec{q}_t = \vec{q}_t{}^* + \vec{r}_t$ for some vector $\vec{r}_t$ such that $\vec{r}_t \cdot \vec{x}_t = 0$ for $i = 1, \ldots, t-1$.

For any $t > 1$, if $\vec{x}_t$ is linearly dependent on $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$, i.e., $\vec{x}_t = \sum_{i=1}^{t-1} \beta_i \vec{x}_i$ for some $\beta_i \in R$, then we have

$$\begin{aligned} q_t \cdot x_t &= (\vec{q}_t{}^* + \vec{r}_t) \cdot \sum_{i=1}^{t-1} \beta_i \vec{x}_i \\ &= \vec{q}_t{}^* \cdot \sum_{i=1}^{t-1} \beta_i \vec{x}_i + \sum_{i=1}^{t-1} \beta_i \vec{r}_t \cdot \vec{x}_i \\ &= \vec{q}_t{}^* \cdot \sum_{i=1}^{t-1} \beta_i \vec{x}_i = \vec{q}_t{}^* \cdot \vec{x}_i \end{aligned}$$

Hence, the algorithm A has the same classification on $\vec{x}_t$ as Rocchio's algorithm does. In other words, if both the algorithm A and Rocchio's algorithm have different classifications on $\vec{x}_t$, then $\vec{x}_t$ must be linearly independent of $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{t-1}$. Note that $\vec{x}_t$ is a counterexample for the hypothesis linear classifier $(\vec{q}_t{}^*, \phi_t)$ of Rocchio's algorithm. The above analysis implies that if $\vec{x}_t$ is not a counterexample for the hypothesis linear classifier $(\vec{q}_t, \phi_t)$ of the algorithm A, then $\vec{x}_t$ must be linearly independent of $\vec{x}_t, \vec{x}_2, \ldots, \vec{x}_{t-1}$. Because there are at most d many linearly independent vectors over the vector space $\{0, \ldots, n-1\}^d$, this can only happen at most d times. This follows that the learning complexity of Rocchio's algorithm is at most $d + l(A)$. For each $t > 1$, the projection $\vec{q}_t{}^*$ can be computed using standard matrix operations in $O(d^2 \log^2 n)$ time, and the above simulation procedure runs at most $d + l(A)$ iterations. Therefore, the time complexity of Rocchio's algorithm is $O(d^2 \log^2 n\,(d + l(A)) + t(A))$.

Maass and Turán (1994) studied online learning of linear classifiers (or half spaces) with equivalence queries. Our adaptive learning model in Definition 2 is the same as their online learning model with equivalence queries. We restate their Theorem 3.3 (p. 393), which was proved using linear programming technique, with our term of adaptive learning from examples in the following:

Theorem 1 (Maass & Turán, 1994): There is an adaptive learning algorithm for learning linear classifiers from examples over the space $\{0, \ldots, n-1\}^d$ with an $O(d^2(\log d + \log n))$ upper bound on its learning complexity. Moreover, the time complexity of the algorithm is polynomial in d and log n.

With the help of Theorem 1 of Maass and Turán (1994) and Lemma 1, we are ready to give a general upper bound on the learning complexity for Rocchio's algorithm.

Theorem 2: The learning complexity of Rocchio's similarity-based relevance feedback algorithm in searching for documents represented by a linear classifier over the d-dimensional discretized vector space $\{0, \ldots, n-1\}^d$ is $O(d + d^2(\log d + \log n))$. Moreover, the time complexity of achieving this upper bound is polynomial in d and log n.

Proof: Let A be the adaptive learning algorithm given by Theorem 1 for learning linear classifiers over the space $\{0, \ldots, n-1\}^d$ from examples. The algorithm A uses linear classifiers as hypotheses. We use the procedure given in the proof of Lemma 1 to allow Rocchio's algorithm to simulate this algorithm A. Then, by Lemma 1 and Theorem 1, the learning complexity of Rocchio's algorithm is $O(d + d^2(\log d + \log n))$, and the time complexity is polynomial in d and log n.

Upper Bounds for Documents Represented by a Monotone Linear Classifier

In this section, we consider the learning complexity of Rocchio's similarity-based relevance feedback algorithm in searching for documents represented by a monotone linear classifier $(\vec{q}, 0)$ over the discretized vector space $\{0, \ldots, n-1\}^d$, where $\vec{q} = (q_1, q_2, \ldots, q_d)$, $q_i \geq 0$, $1 \leq i \leq d$. For a monotone linear classifier $(\vec{q}, 0)$, any $\vec{x} \in \{0, \ldots, n-1\}^d$ is classified as relevant if

$$\vec{q} \cdot \vec{x} = q_1 x_1 + q_2 x_2 + \cdots + q_d x_d > 0, \tag{2}$$

or is classified as irrelevant otherwise. The efficient learnability of monotone linear classifiers has been extensively studied in machine learning (for example, Littlestone, 1988). Monotone linear classifiers are straightforward extensions of monotone disjunctions of relevant features (or attributes). Although very simple in format, monotone disjunctions are very common ways of expressing search queries, especially in the case of Web search. All existing popular search engines support *OR combinations* (or monotone disjunctions) of keywords as search query formations.

Theorem 3: The learning complexity of Rocchio's similarity-based relevance feedback algorithm in searching for documents represented by a monotone linear classifier $(\vec{q}, 0)$ over the discretized vector space $\{0, \dots, n-1\}^d$ is at most $1 + 2k(n-1)(\log d + \log(n-1))$. Here, k is the number of nonzero components of $\vec{q}$.

We postpone the proof to the end of this section, but first give the following corollary, which resembles the bound of the well-known algorithm Winnow1 (Littlestone, 1988).

Corollary 1: The learning complexity of Rocchio's similarity-based relevance feedback algorithm in searching for documents represented by a monotone linear classifier $(\vec{q}, 0)$ over the Boolean vector space $\{0, 1\}^d$ is at most $1 + 2k \log d$. Here, k is the number of nonzero components of $\vec{q}$.

We now extend the multiplicative query updating technique developed by Littlestone (1988) for learning monotone linear classifiers over the Boolean vector space $\{0, 1\}^d$ to the discretized vector space $\{0, \dots, n-1\}^d$ to search for documents represented by a monotone linear classifier $(\vec{q}, 0)$ satisfying expression (2). It is easy to see that additive query updating yields some *mild* improvement on the hypothetical query vector towards the target linear classifier, whereas multiplicative query updating can yield dramatic improvements so that the hypothetical query vector can be moved towards the target in a much faster pace. The idea is that when a document $\vec{x}$ is judged by the user as relevance feedback, for the ith component x_i of $\vec{x}$ corresponding the ith index term, the value of the ith component of the hypothetical query vector should be boosted by a multiplicative factor and x_i as well. We present the multiplicative adaptive learning algorithm, denoted as MAL, in the following:

Algorithm MAL:

```
(i) Inputs:
        $\vec{q}_1 = \vec{1} = (1, 1, \dots, 1)$, the initial query vector.
        $\alpha$: the query updating factor.
        $\phi \geq 0$, the classification threshold.
(ii) Set $t = 1$.
(iii) Classify and rank documents with the linear classifier $(\vec{q}_1, \phi)$.
(iv) While (the user judged the relevance of a document $\vec{x}$) do {
        For $i = 1, \dots, d$, do {
            /* $\vec{q}_t = (q_{1,t}, \dots, q_{d,t})$, $\vec{x} = (x_1, \dots, x_d)$. */
            If ($x_i \neq 0$) {
                If ($\vec{x}$ is relevant)   /* promotion step */
                    Set $q_{i,t+1} = \alpha^{\frac{x_i}{n-1}} q_{i,t}$
                Else   /* demotion step */
                    Set $q_{i,t+1} = 0$
            } Else
                Set $q_{i,t+1} = q_{i,t}$
        }
    }
(v) If no documents were judged in the $t$th step, then stop.
    Otherwise, let $t = t + 1$ and go to step (iv).
```

We observe that algorithm MAL differs from algorithm Winnow1 (Littlestone, 1988) at the promotion step as follows: MAL uses $\alpha^{\frac{x_i}{n-1}} q_{i,t}$ to update $q_{i,t+1}$, whereas algorithm Winnow1 uses $\alpha\ q_{i,t}$. We now analyze the performance of algorithm MAL when it is used to search for documents represented by a monotone linear classifier $(\vec{q}, 0)$ satisfying Expression (2).

Lemma 2: Let u denote the total number of promotions that algorithm MAL needs to search for documents represented by a monotone linear classifier $(\vec{q}, 0)$. If $\alpha > n - 1$, then, $u \leq k \frac{\log \phi}{\log \frac{\alpha}{n-1}}$. Here, k is the number of nonzero components $q_i > 0$.

Proof: Without loss of generality, we may further assume that the k nonzero components of $\vec{q}$ are $q_1, q_2, \dots, q_k$ and all the other components are zero. When a promotion occurs at step t, a relevant document $\vec{x}$ is given to the algorithm as a counterexample to its current classification. That is, there must be some i with $1 \leq i \leq k$ such that $x_i \geq 1$. This means that the ith component $q_{i,t}$ of the query vector $\vec{q}_t$ will be promoted to

$$q_{i,t+1} = \alpha^{\frac{x_i}{n-1}} q_{i,t} \geq \alpha^{\frac{1}{n-1}} q_{i,t},$$

because $1 \leq x_{\mathrm{i}} \leq n - 1$. By Expression (2), $q_{i,t}$ will never be demoted. Because $q_{i,1} = 1$, this follows that $q_{i,t}$ can be promoted at most

$$\frac{\log \phi}{\log \frac{\alpha}{n-1}} \tag{3}$$

times. Because each promotion yields a promotion for at least one $q_{i,t}$ for $1 \leq i \leq k$, the total number of promotions u is at most k times the value given in Expression (3).

Lemma 3: Let T denote the learning complexity of the algorithm MAL in searching for documents represented a monotone linear classifier $(\vec{q}, 0)$ over the discretized vector space $\{0, \dots, n-1\}^d$. Let k denote the number of nonzero components of $\vec{q}$. Suppose $\alpha > n - 1$. Then,

$$T \leq \frac{d(n-1)}{\phi} + k\alpha \frac{\log \phi}{\log \frac{\alpha}{n-1}}. \tag{4}$$

Proof: Without loss of generality, we may assume again that the k nonzero components of $\vec{q}$ are $q_1, q_2, \dots, q_k$ and all the other components are zero. We estimate the sum of $\sum_{i=1}^{d} q_{i,t}$. Let u and v be the number of promotion steps and the number of demotion steps that occurred during the learning process, respectively. For a promotion at step t with respect to a relevant document $\vec{x}$ judged by the user, for $i = 1, \ldots, d$, since $0 \leq x_i \leq n - 1$, we have

$$q_{i,t+1} = \begin{cases} q_{i,t} = q_{i,t} + (\alpha - 1)\dfrac{x_i}{n-1}q_{i,t}, & \text{if } x_i = 0, \\ \alpha\dfrac{x_i}{n-1}q_{i,t} \leq q_{i,t} + (\alpha - 1)\dfrac{x_i}{n-1}q_{i,t}, & \text{if } x_i \neq 0. \end{cases}$$

Thus,

$$q_{i,t+1} \leq q_{i,t} + (\alpha - 1)\frac{x_i}{n-1}q_{i,t}.$$

Because a promotion occurs only when

$$\vec{q}_t \cdot \vec{x} = \sum_{i=1}^{d} q_{i,t}x_i < \phi,$$

we have

$$\begin{aligned} \sum_{i=1}^{d} q_{i,t+1} &\leq \sum_{i=1}^{d} q_{i,t} + \frac{\alpha - 1}{n-1}\sum_{i=1}^{d} q_{i,t}x_i \\ &\leq \sum_{i=1}^{d} q_{i,t} + \frac{\alpha - 1}{n-1}\phi. \end{aligned} \tag{5}$$

For a demotion at step t with respect to an irrelevant document $\vec{x}$ judged by the user, for $i = 1, \ldots, d$, since again $0 \leq x_i \leq n - 1$, we have

$$q_{i,t+1} = \begin{cases} q_{i,t} = q_{i,t} - \dfrac{x_i}{n-1}q_{i,t}, & \text{if } x_i = 0, \\ 0 = q_{i,t} - q_{i,t} \leq q_{i,t} - \dfrac{x_i}{n-1}q_{i,t}, & \text{if } x_i \neq 0. \end{cases}$$

Thus, we have

$$q_{i,t+1} \leq q_{i,t} - \frac{x_i}{n-1}q_{i,t}.$$

Because a demotion occurs only when

$$\vec{q}_t \cdot \vec{x} = \sum_{i=1}^{d} q_{i,t}x_i \geq \phi,$$

we have

$$\begin{aligned} \sum_{i=1}^{d} q_{i,t+1} &\leq \sum_{i=1}^{d} q_{i,t} - \frac{1}{n-1}\sum_{i=1}^{d} q_{i,t}x_i \\ &\leq \sum_{i=1}^{d} q_{i,t} - \frac{1}{n-1}\phi. \end{aligned} \tag{6}$$

Note that the initial query vector $\vec{q}_1 = \vec{1} = (1, 1, \ldots, 1)$. By Expressions (5) and (6), after u promotions and v demotions,

$$\sum_{i=1}^{d} q_{i,t+1} \leq d + \frac{(\alpha - 1)}{n-1}\phi u - \frac{1}{n-1}\phi v. \tag{7}$$

Because at any step the components of the query are never negative, it follows from Expression (7) that

$$v \leq \frac{d(n-1)}{\phi} + (\alpha - 1)u. \tag{8}$$

It follows from Lemma 2 and Expression (8) that the total number of promotions and demotions, i.e., the learning complexity T, is bounded by

$$T \leq v + u \leq \frac{d(n-1)}{\phi} + k\alpha\frac{\log \phi}{\log \dfrac{\alpha}{n-1}}.$$

This completes our proof.

Proof of Theorem 3: It follows directly from the above Lemma 3 and Lemma 1 in the previous section with the choices of $\phi = d(n - 1)$ and $\alpha = 2(n - 1)$.

Lower Bounds

Maass and Turán (1994) have derived the following lower bound on the number of different linear classifiers over the discretized vector space $\{0, \ldots, n - 1\}^d$:

Proposition 1 (Maass & Turán, 1994): The number of different linear classifiers over the discretized vector space $\{0, \ldots, n - 1\}^d$ is at least $n^{\binom{d}{2}}(n - 1)^d$. Based on the above lower bound on the number of linear classifiers and the binary decision tree technique devised by Littlestone (1988), they obtained an $\Omega(\binom{d}{2}\log n)$ lower bound for any adaptive (online) learning algorithm for learning linear classifier over $\{0, \ldots, n - 1\}^d$. This implies the following corollary.

Corollary 2: The learning complexity of Rocchio's algorithm in searching for documents represented by a linear classifier over the discretized vector space $\{0, \ldots, n - 1\}^d$ is at least $\Omega(\binom{d}{2}\log n)$. In particular, in the Boolean vector space $\{0, 1\}^d$, the lower bound is $\Omega(d^2)$.

Remark 1: The above lower bound does not apply to the case of searching for documents represented by a monotone linear classifier over the discretized vector space $\{0, \ldots, n - 1\}^d$ because there are fewer monotone linear classifiers over $\{0, \ldots, n - 1\}^d$ than general linear classifiers, so that the $\Omega(\binom{d}{2}\log n)$ lower bound on the number of linear classifiers in general does not hold for monotone linear classifiers. In particular, there are at most $\binom{d}{k}$ monotone disjunctions $x_{i_1} \vee \cdots \vee x_{i_k}$ over the Boolean vector space $\{0, 1\}^d$. We can only derive an $\Omega(k \log d)$ lower bound for searching for documents represented by a monotone disjunction of k Boolean relevant features. When k is a constant, this lower bound becomes $\Omega(\log d)$. In Chen and Zhu (2002), an $\Omega(d)$ lower bound is obtained for monotone disjunctions of k relevant features over the Boolean vector space $\{0, 1\}^d$.

We give an example to compare Corollary 2 with the linear lower bound obtained in (Chen & Zhu, 2002). To simplify presentation, we assume that the constants in these lower bounds are 1. We consider a typical Web searching scenario with $d = 300$, $k = 4$, $n = 20$. In this case, it follows from Proposition 1 that there are at least $20^{150\times 299} \times 19^{300}$ many different linear classifiers over the discretized vector space $\{0, \ldots, 19\}^{300}$. By Corollary 1, Rocchio's algorithm needs to have at least 193,839 many examples or documents judged by the user as relevance feedback to learn, in the worst case, a set of documents represented by a linear classifier over $\{0, \ldots, 19\}^{300}$. Similarly, Rocchio's algorithm needs to have at least 44,850 many examples or documents judged by the user as relevance feedback to learn, in the worst case, a set of documents represented by a linear classifier over the Boolean vector space $\{0, 1\}^{300}$. When only monotone disjunction $x_{i_1} \vee \cdots \vee x_{i_k}$ of k relevant features or index terms are considered over the Boolean vector space $\{0, 1\}^{300}$, there are at most 330,791,175 many different monotone disjunctions, which are far fewer than $20^{150\times 299} \times 19^{300}$. Following a similar approach to Corollary 1, we have that Rocchio's algorithm needs to have at least $k \log n = 4 \log 20 \approx 18$ many examples or documents judged by the user as relevance feedback to learn, in the worst case, a set of documents represented by a monotone disjunction of k relevant features or index terms over the Boolean vector space $\{0, 1\}^{300}$, whereas the linear lower bound obtained in Chen and Zhu (2002) implies that Rocchio's algorithm needs to have at least 300 many examples or documents judged by the user as relevance feedback to do the same. Obviously, the linear lower bound in Chen and Zhu is stronger than the lower obtained here for searching for a set of documents represented by a monotone disjunction of k relevant features or index terms over the Boolean vector space. However, the work in Chen and Zhu cannot be generalized to the discretized vector space, or to general linear classifiers.

In practice, a fixed query updating factor α is used for Rocchio's algorithm. At any step $t \geq 1$, for $1 \leq i \leq d$, the i-component of the query vector $\vec{q}_{t+1}$ is updated with respect to the counterexample $\vec{x}_t = (x_{1,t}, \ldots, x_{d,t})$ as follows: $q_{i,t+1} = q_{i,t} + \alpha\, x_{i,t}$ if $\vec{x}_t$ is relevance, otherwise $q_{i,t+1} = q_{i,t} - \alpha\, x_{i,t}$. In general, the classification threshold can be reviewed as an additional variable, it can be updated as $\phi_{t+1} = \phi_t + \alpha$ if $\vec{x}_t$ is relevance, otherwise $\phi_{t+1} = \phi_t - \alpha$. Following the approach for deriving a lower bound for *k-bound* learning algorithm in Maass and Turán (1994), we have the following theorem.

Theorem 4. If a fixed query updating factor is used, then the learning complexity of Rocchio's similarity-based relevance feedback algorithm in searching for documents represented by a linear classifier over the Boolean vector space $\{0, 1\}^d$ is at least $2^{\Omega(d)}$.

Proof: Note that at any step $t \geq 1$, the counterexample $\vec{x}_t = (x_{1,t}, \ldots, x_{d,t})$ to the query vector $\vec{q}_t$ is a Boolean vector in $\{0, 1\}^d$. As analyzed above, for $1 \leq i \leq d$, the ith component $q_{i,t+1}$ of the query vector $\vec{q}_{t+1}$ can be updated as either $q_{i,t}$, or $q_{i,t} + \alpha$, or $q_{i,t} - \alpha$, depending on whether $\vec{x}_t$ is relevant or not and whether x_i is zero or one. By iteration from Expression (1), $q_{i,t+1}$ is obtained from $q_{\mathrm{i},1}$ with t many operations, each of which is one of three types of $+0$, $+\alpha$, and $-\alpha$. The order of these operations do not affect the value of $q_{i,t+1}$. The value of $q_{i,t+1}$ is determined by the number of each type of operations involved. Thus, there are at most t^3 many possible values for $q_{i,t+1}$. This means that there are at most t^{3d} many possible choices for q_{t+1}. Similarly, ϕ_{t+1} can be updated as either $\phi_t + \alpha$ or $\phi_t - \alpha$, implying that there are at most t^2 many possible values for ϕ_{t+1}. Therefore, at step t, Rocchio's algorithm with a fixed query updating factor can generate at most $t^{3(d+1)}$ many possible hypotheses. By Proposition 1, to search for sets of documents represented by any linear classifier over the Boolean vector space $\{0, 1\}^d$, we must have

$$t^{3(d+1)} \geq 2^{\binom{d}{2}}.$$

Hence, $t \geq 2^{\Omega(d)}$.

Remark 2: As commented in Remark 1, the lower bound in Theorem 4 does not apply to the case of searching for documents represented by a monotone linear classifier over discretized vector space $\{0, \ldots, n-1\}^d$, because there are fewer monotone linear classifiers over $\{0, \ldots, n-1\}^d$ than general linear classifiers, so that the $\Omega(\binom{d}{2} \log n)$ lower bound on the number of linear classifiers in general does not hold for monotone linear classifiers. In Chen and Fu (2005), an $\Omega(k(n-k))$ lower bound is obtained for Rocchio's algorithm in searching for documents represented by monotone disjunctions of k relevant features over $\{0, 1\}^d$.

We compare Theorem 4 with the lower bounds obtained in Chen and Zhu (2002) and Chen and Fu (2005) in detail in the following. To simplify presentation, we assume that the constants in these lower bounds are 1. We consider a typical Web-searching scenario with $d = 300$, $k = 4$, $n = 20$. We also consider that a fixed query updating factor is used by Rocchio's algorithm. It follows from Theorem 4 that Rocchio's algorithm needs to have at least 2^{300} many examples or documents judged by the user as relevance feedback to learn, in the worst case, a set of documents represented by a linear classifier over the Boolean vector space $\{0, 1\}^{300}$. However, this result does not apply to the case of searching for documents represented by monotone disjunctions $x_{i_1} \vee \cdots \vee x_{i_k}$ of k relevant features or index terms over the Boolean vector space $\{0, 1\}^{300}$. As discussed before, the lower bounds, as obtained by Chen and Zhu and Chen and Fu, apply to the case of monotone disjunctions of k relevant features or index terms. In particular, the linear lower bound obtained by Chen and Zhu implies that Rocchio's algorithm needs to have at least 300 examples or documents judged by the user as relevance feedback in searching for a set of documents represented by a monotone disjunction $x_{i_1} \vee \cdots \vee x_{i_k}$ of k relevant features or index terms over the Boolean vector space $\{0, 1\}^{300}$, whereas the lower bound obtained by Chen and Fu implies that Rocchio's algorithm needs to have at least $k(n - k) = 4 \times 296 = 1{,}184$ examples or documents judged by the user as relevance feedback to do the same. Obviously, the lower bound obtained by Chen and Fu is stronger than the lower bound obtained by Chen and Zhu in the case of monotone disjunctions of k relevant features when a fixed query updating factor is used.

Unfortunately, the proof of Theorem 4 cannot be generalized to the discretized vector space $\{0, \ldots, n - 1\}^d$. In this case, $q_{i,t+1}$ is obtained from $q_{i,1}$ with k many operations, each of which is one of three types of operations $+0$, $+\alpha\, x_{i,t}$, and $-\alpha x_{i,t}$. Because $x_{i,t}$ in general may not be 1 or 0, the value of $q_{i,t+1}$ is determined by not only the number of each type of operations involved, but also the order of these operations. However, we have the following lower bound.

Theorem 5: Suppose that the query updating factors used by Rocchio's similarity-based algorithm are bounded by $O(n^c)$ from some constant $c \geq 0$ during its process of searching for documents represented by a linear classifier over the discretized vector space $\{0, \ldots, n - 1\}^d$. Then, the learning complexity of Rocchio's algorithm is at least $\Omega(n^{d-1-c}/(n - 1))$.

Proof: It is proved in Hampson and Volper (1990) that there are linear classifiers $(\vec{q}, 0)$ over $\{0, \ldots, n - 1\}^d$ such that $q_i = \theta(n^{d-1})$ for some i, $1 \leq i \leq d$. At each step $t \geq 1$, by the given condition of the theorem, Rocchio's algorithm can update its query vector by a magnitude of at most $O(n^c(n - 1))$. Hence, the algorithm needs at least $\Omega(n^{d-1-c}/(n - 1))$ steps to learn $q_i = \theta(n^{d-1})$.

Remark 3: The exponential lower bounds obtained in Theorems 4 and 5 do not contradict the $O(d + d^2(\log d + \log n))$ upper bound obtained in Theorem 2 and the $1 + 2k(n - 1)(\log d + \log(n - 1))$ upper bound obtained in Theorem 3. In Theorems 4 and 5, the query updating factors are either fixed or bounded by $O(n^c)$; there are no such requirements in Theorems 2 and 3. In reality, when computing the projection of a query vector onto a linear subspace spanned by a list of counterexamples in the proof of Lemma 1, exponential query updating factors may occur. The multiplicative query updating technique used to prove Theorem 3 aims at boosting the related components of the query vector at a fast, possibly exponential, pace.

Concluding Remarks

It would be very interesting to analyze the average-case learning complexity of Rocchio's algorithm. We feel that this problem is very challenging because any nontrivial average case analysis will reply on realistic models of document distribution, index term distribution, and the user preference distribution as well. We feel that it is not easy to model those distributions, or to analyze the complexity under those distributions. The probabilistic corpus model proposed in Papadimitriou, Raghavan, and Tamaki (2000) may shed some light on this problem.

Acknowledgments

Zhixiang Chen's research was supported in part by NSF CNS-0521585. Part of Bin Fu's work was done when he was on the computer science faculty at the University of New Orleans and was also affiliated with the Research Institute for Children, 200 Henry Clay Avenue, New Orleans, LA 70118. Bin Fu's research was supported by Louisiana Board of Regents under contract number LEQSF (2004–07)-RD-A-35.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (Eds.). (1999). Modern information retrieval. Essex, UK: Addison-Wesley.

Bshouty, N., Chen, Z., Decatur, S., & Homer, S. (1995). On the learnability of Z_n-DNF formulas. In P. Auer & R. Merr (Eds.), Proceedings of the 18th Annual Conference on Computational Learning Theory, Lecture Notes in Computer Science 3559 (pp. 198–205). Berlin: Springer.

Chen, Z. (2001). Multiplicative adaptive algorithms for user preference retrieval. In J. Wang (Ed.), Proceedings of the Seventh Annual International Computing and Combinatorics Conference, Lecture Notes in Computer Science 2108 (pp. 540–549). Berlin: Springer.

Chen, Z. (2004). Multiplicative adaptive user preference retrieval and its applications to web search. In G. Zhang, A. Kandel, T. Lin, & Y. Yao (Eds.), Computational web intelligence: Intelligent technology for web applications (pp. 303–328). Singapore: World Scientific.

Chen, Z., & Fu, B. (2005). A quadratic lower bound for Rocchio's similarity-based relevance feedback algorithm. In L. Wang (Ed.), Proceedings of the Eleventh Annual International Computing and Combinatorics Conference, Lecture Notes in Computer Science 3595 (pp. 955–964). Berlin: Springer.

Chen, Z., & Zhu, B. (2002). Some formal analysis of Rocchio's similarity-based relevance feedback algorithm. Information Retrieval, 5, 61–86.

Frakes, W., & Baeza-Yates, R. (Ed.). (1992). Information retrieval: Data structures and algorithms. Englewood Cliffs, NJ: Prentice Hall.

Ide, E. (1971a). Interactive search strategies and dynamic file organization in information retrieval. In G. Salton (Ed.), The smart system—Experiments in automatic document processing (pp. 373–393). Englewood Cliffs, NJ: Prentice-Hall.

Ide, E. (1971b). New experiments in relevance feedback. In G. Salton (Ed.), The smart system—Experiments in automatic document processing (pp. 337–354). Englewood Cliffs, NJ: Prentice-Hall.

Hampson, S., & Volper, D. (1990). Representing and learning Boolean functions of multivalued features. IEEE Transactions on Systems, Man, and Cybernetics, 20, 67–80.

Kivinen, J., Warmuth, M., & Auer, P. (1997). The perceptron algorithm vs. Winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. Artificial Intelligence, 1/2, 325–343.

Lewis, D. (1991). Learning in intelligent information retrieval. In L. Birnbaum & G. Collins (Ed.), Proceedings of the Eighth International Workshop on Machine Learning (235–239). San Francisco: Morgan Kaufmann.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning, 2, 285–318.

Maass, W., & Turán, G. (1994). How fast can a threshold gate learn? Computational Learning Theory and Natural Learning Systems, 1, 381–414.

Papadimitriou, C., Raghavan, P., & Tamaki, H. (2000). Latent semantic indexing: A probabilistic analysis. Journal of Computer and System Science, 61(2), 217–235.

Raghavan, V., & Wong, S. (1986). A critical analysis of the vector space model for information retrieval. Journal of the American Society for Information Science and Technology, 37(5), 279–287.

Rocchio, J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), The smart retrieval system—Experiments in automatic document processing (pp. 313–323). Englewood Cliffs, NJ: Prentice-Hall.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–407.

Salton, G. (Ed.). (1989). Automatic text processing: The transformation, analysis, and retrieval of information by computer. Boston: Addison-Wesley.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science and Technology, 41(4), 288–297.

Salton, G., Wong, S., & Yang, C. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620.

van Vijsbergen, C.J. (1979). Information retrieval. Boston: Butterworths.

Wong, S., Yao, Y., & Bollmann, P. (1988). Linear structures in information retrieval. In N. Belkin, & C.J. van Rijsbergen (Eds.), Proceedings of the 1988 ACM-SIGIR Conference on Research and Development in Information Retrieval (pp. 219–232). New York: ACM Press.