Learning in Search for World Wide Web Documents with Queries

Zhixiang Chen and Xiannong Meng and Richard Fowler
Department of Computer Science, University of Texas – Pan American
1201 West University Drive, Edinburg, TX 78539–2999, USA
{chen, meng}@cs.panam.edu, fowler@panam.edu

Abstract

In this paper we study the problem of designing intelligent search engines with machine learning techniques. An intelligent search engine should be able to interact with the user and learn to refine its search process and find the desired documents for the user with as few queries as possible. In spite of the growing popularity of existing search engines, little is known about the following fundamental question: How many queries are needed to search for a collection of documents represented by a disjunction (or a conjunction) of attributes? Algorithmic solutions to the above question not only have theoretical significance but may also be used in implementing search engines. We consider that web documents can be represented by vectors of boolean attributes. We design several algorithms for searching any collection of documents represented by a disjunction (or a conjunction) of relevant attributes with the help of membership queries (or equivalence queries).

Introduction

The World Wide Web is a huge distributed heterogeneous database. Its enormous size makes it necessary to design innovative strategies and algorithms for hypertext and image document search. Nowadays there are a number of search engines for people to search for their desired Web documents. Each of the existing search engines has a unique interface and an index database covering a different portion of the Web. They have proved both useful and popular. In general, when using a search engine an average user needs to repeatedly refine queries as he or she does not have enough knowledge to formulate the queries precisely. Usually the engine returns tremendously many URL's that are irrelevant, forcing the user to manually sift through a long list to locate the desired documents.

In this paper, we study the problem of designing efficient search engines within an index database. A search engine should use as little relevance feedback from a user as possible. This problem is challenging because of two major difficulties: communication between user and search engine, and the usage of relevance feedback. Although a user clearly (at the common sense level) knows what kinds of documents he or she wants, it is difficult, if not impossible, for the user to inform the search engine what he or she wants in an easy way so that the search engine understands. Even when disjunctions or conjunctions of keywords are chosen as the way of expressing the communication, the user may not know what set of keywords and what types of conjunctions or disjunctions should be used to precisely define the collection of the documents desired. On the other hand, having received relevance feedback from the user, the search engine needs to find an efficient way to use the feedback so that it can effectively improve the result of its next search.

Using a search engine is much like a dialogue between the user and the engine. The user inputs a query to the engine, the engine uses the query to search the index database and returns a list of URL's. Then the user provides the engine relevance feedback. The engine uses the feedback to improve its next search and returns a refined list of URL's; and the dialogue ends when the engine finds the desired documents. Note that conceptually a query entered by the user can be understood as the logical expression of the collection of desired documents. A list of URL's returned by the engine can be interpreted as an approximation to the collection of the desired documents. This type of scenario is essentially the same as the process of online learning with queries (Angluin 1987; Littlestone 1988), when the user is interpreted as a teacher and the engine as a learner. In online learning with queries (Angluin 1987; Littlestone 1988), the goal of the learner is to find an unknown target concept chosen by the teacher. The learner can present hypotheses to the teacher. If a hypothesis is the same as the target, then the teacher tells so and thus concludes the learning process. Otherwise, the teacher provides the learner with a counterexample as feedback. To make a response to any given example, the teacher answers "yes" or "no" depending on whether the example is in the target. Because of this similarity, we use an online learning approach to study the problem of designing efficient search engines. Our study is based on the two following assumptions:

Membership Query Assumption. *During a*

search process, given any particular web document, a user clearly knows whether it is desired, i.e., whether it is in the collection of web documents the user wants to find.

Target Concept Assumption. *When a user uses a search engine to search for a collection of web documents, the collection must be uniquely determined. The user knows what the collection is (in a way much like defining a set by giving properties of set elements), though the user may not know how to precisely define the collection with simple logical statements such as disjunctions or conjunctions of keywords.*

These two assumptions reflect the reality of Web search for average users. It would be meaningless to carry out a search when one or both of the assumptions are not true. One must distinguish between the two assumptions and the case in which a user may make errors in inputing queries and providing feedback. The latter case should be studied in the future. In this paper, we use machine learning techniques to design intelligent search algorithms. Since existing search engines support queries represented by disjunctions or conjunctions of attributes, we want to know how many queries are sufficient for a search engine to find a collection of documents represented by a disjunction (or a conjunction) of attributes. Since an average user may not have the patience to try more than 50 queries at all, we need to find some "practically tolerable query complexity" (for example, around one or two dozens of queries) demanded by a user. Unfortunately, well-established efficient machine learning algorithms such as those in (Bshouty, Chen, & Homer 1994; Chen 1995; Chen & Homer 1996; Chen & Maass 1994) are not suitable to design intelligent search engines, simply because the query complexity bounds are too high, though polynomial. There are successful applications of machine learning techniques in some aspects of Web mining (Etzioni 1997) (for example, (Armstrong *et al.* 1995; Perkowitz *et al.* 1997)). But little is known in literature about the "practically tolerable query complexity" for search engines in spite of their growing popularity.

Based on the online learning model with queries (Angluin 1987; Littlestone 1988), we show that any collection of web documents represented by disjunctions (or conjunctions) of k relevant attributes can be exactly identified with at most $k \log_2 n$ membership queries, or with at most $1.885k \log_2 \frac{n}{k} - k$ equivalence queries, where n is the total number of attributes used to represent web documents.

As estimated in (Lawrence & Giles 1998) currently there are at least 320 million indexable web documents over the Internet and each search engine covers only a portion of the Web. Since $2^{29} = 536,870,912 > 320,000,000$, a vector space of dimension 29 is capable of representing indexable web documents for a search engine. For small k, for example, $k = 5$, our algorithms show that any collection of web documents represented by disjunctions (or conjunctions) of 5 attributes can be exactly identified with at most 25 membership queries, or with at most 24 equivalence queries. In fact, the number of queries can be reduced if we relax the exact identification requirement to approximate identification. One method for approximate identification is to combine our learning algorithms with the vector similarity ranking techniques (Salton 1989). In one of our ongoing research projects, we study how to combine our algorithms with personalization (Meng & Chen 1999) to further reduce the number of queries required by a search engine.

This paper is organized as follows: In section 2 we discuss the well-known vector space model and use it to represent web documents. In section 3, we introduce the online learning model with queries and discuss how to use this model to solve the problem of web search. In sections 4 and 5 we design algorithms to search for web documents with membership and equivalence queries. In section 6, we discuss the issues of implementing our algorithms and also list problems for future investigation.

The Vector Space Model and the Relevance Feedback

There are various mathematical models for representing information retrieval systems. The vector space model among those is perhaps the simplest to use and in some ways the most productive (Salton 1989). In the vector space model, each document is represented as an n-dimensional vector $x = (x_1, \ldots, x_n)$ of n attributes. Attributes may be boolean-valued. In other cases, we can allow an attribute to have nonnegative values to reflect its importance for the documents. For example, when we can choose a set of n keywords as attributes $\{K_1, \ldots, K_n\}$, the vector $(x_1, \ldots, x_n)$ representing a document is defined as follows: The presence of K_i in the document is indicated by $x_i = 1$ and the absence is indicated by $x_i = 0$. In the vector space model, a query Q is also represented by an n-dimensional vector $(q_1, \ldots, q_n)$, where the value of q_i indicates the absence (or importance) of the i-th attribute. The indexing procedure is the process of building attribute vectors for all documents. Once this procedure is done, information retrieval can be performed over the vector space as follows: Given any query $Q = (q_1, \ldots, q_n)$, find and display all documents $D = (x_1, \ldots, x_n)$ such that Q and D are similar. There are many ways to measure the similarity of two vectors (Salton 1989). For example, the value of the inner product of two vectors may be used as the similarity. The similarity can also be used as a means to rank documents when displaying them to the user.

As pointed out in (Salton 1989), one of the most important and difficult operations in information retrieval is generating useful query statements that can extract the desired documents for the user and exclude

the remainder. Since usually an ideal query representation cannot be generated without knowing a great deal about the composition of the collection of the desired documents, it is customary to conduct searches iteratively, first operating with a tentative query formation and then improving formulations for subsequent searches based on evaluations of the previously retrieved materials. One way for automatically generating the improved query formulation is the well-known relevance feedback process.

In the the relevance feedback process the user indicates the system with relevant documents among those retrieved. The system uses the feedback from the user to reformulate the query vector and starts the search from the beginning. The usual way of reformulating the query vector is by adding the weighted vectors of the relevant documents to the query vector and by subtracting the vectors of the irrelevant documents.

Web Searching and Online Learning

As estimated in (Lawrence & Giles 1998), currently there are at least 320 million indexable web documents over the Web. The growth of the size of the Web is rapid, and may be increased 1000% in a few years. Indeed, the Web provides all people in the world with useful information. But the enormous size of the Web and its unstructured nature make the task of searching for desired and relevant information very challenging (see (Etzioni 1997) for a good introduction to web mining). Search engines such as Yahoo, Altavisa, Lycos, etc., help people in information discovery. However, when using any of the existing search engines, the user receives a long list of irrelevant, outdated, or unavailable URL's of documents. Users are forced to manually sift through a long list to search for useful information.

The interaction between search engine and user is much like the process of online learning with queries. In the online learning model (Angluin 1987; Littlestone 1988) with equivalence and membership queries, the goal of a learner for a concept class $\mathbf{C}$ over the domain $\mathbf{Z}$ is to learn any unknown target concept $c_t \in \mathbf{C}$ that has been fixed by a teacher. In order to obtain information about c_t, the learner can ask the teacher equivalence queries by proposing hypotheses h from a fixed hypothesis space $\mathbf{H}$ over $\mathbf{Z}$ with $\mathbf{C} \subseteq \mathbf{H}$. If $h = c_t$, then the teacher says "*yes*", so the learner succeeds. If $h \neq c_t$, then the teacher responds with a counterexample x in $(c_t - h) \cup (h - c_t)$, for some $x \in \mathbf{Z}$. x is called a positive counterexample if it is in c_t and a negative counterexample otherwise. The learner can also ask membership queries by presenting examples in the domain to the teacher. For any example x, the teacher says "*yes*" if it is in c_t, and "*no*" otherwise. Each new hypothesis issued by the learner may depend on the earlier hypotheses and the examples observed so far. A learner exactly learns $\mathbf{C}$, if for any target concept $c_t \in \mathbf{C}$, it can learn c_t. We say that a class $\mathbf{C}$ is polynomial time learnable if there is a learner that exactly learns any target concept in $\mathbf{C}$ and runs in time polynomially in the size of the domain, the size of the target concept, and the size of the largest example observed during its learning process.

We now consider how to use the online learning model to approach the problem of web search. We use the vector space model described in the previous section to represent documents. Here we only consider boolean-valued vectors. For any positive integer $n \geq 1$, let V_n denote the n-dimensional boolean vector space $\{0,1\}^n$. V_n is our domain. A target concept is a subset of V_n. The learner is the search engine. The teacher is the user. The goal of the search engine is to find the target concept (that is, a collection of the desired documents) with minimal number of queries and in polynomial times.

In the online learning model, the target concept is chosen by the teacher. Moreover, the teacher knows whether any given hypothesis is the same as (or logically equivalent to) the target concept, and whether any given example is in the target. Now, think about the role played by a *reasonable* user of a search engine. When a user starts a search using the search engine, the user knows whether any given Web page shown on the screen in front is what the user needs. That is, the *Membership Query Assumption* is true for the user, thus the user is able to answer membership queries. On the other hand, under the *Target Concept Assumption*, although the user knows (and hence chosen the target collection of desired documents), he in general does not know any documents in the target collection and this is what the user wants to search for. This means that even if the *Target Concept Assumption* is true, the user may not be able to answer equivalence queries. However, as in practice, the search engine can display a long list of URL's to the user as a hypothesis for the target collection. The user can read the list (or can even get the actual pages linked by some URL's), so the user can tell whether the list of URL's is the collection of the desired documents. If so, the user can respond "yes" to the search engine to stop the search, otherwise, the user can inform the engine of one or more URL's that link to the desired or undesired documents, thus providing counterexamples to the search engine.

Another issue we should consider is that in practice there may be vectors in the domain without actual corresponding web documents. To solve this problem, we can build dummy web pages for all such vectors. A simple dummy web page for such a vector could be a list of all terms relevant to all the one-bits of the vector. When the search engine uses such a vector to make a membership query to the user, the search engine displays the dummy page, and the user should be able to answer whether the page is desired or not.

Definition 3.1. *Given any document d, let*

$v_d(x_1, \ldots, x_n)$ *denote its vector representation. Define*

$$md[x_{i_1}, \ldots, x_{i_k}] = \{d | v_d(x_1, \ldots, x_n) \Longrightarrow x_{i_1} \vee \cdots \vee x_{i_k}\},$$

$$nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}] = \{d | v_d(x_1, \ldots, x_n) \Longrightarrow x_{i_1} \vee \cdots \vee x_{i_l} \vee \bar{x}_{j_1} \vee \cdots \vee \bar{x}_{j_k},\ l + m = k\},$$

$$mc[x_{i_1}, \ldots, x_{i_k}] = \{d | v_d(x_1, \ldots, x_n) \Longrightarrow x_{i_1} \wedge \cdots \wedge x_{i_k}\},$$

$$nc[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}] = \{d | v_d(x_1, \ldots, x_n) \Longrightarrow x_{i_1} \wedge \cdots \wedge x_{i_l} \wedge \bar{x}_{j_1} \wedge \cdots \wedge \bar{x}_{j_m},\ l + m = k\}.$$

In other words, $md[x_{i_1}, \ldots, x_{i_k}]$ is the collection of documents whose vectors satisfy the monotone disjunction of $x_{i_1}, \ldots, x_{i_k}$; $nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}]$ is the collection of documents whose vectors satisfy the non-monotone disjunction of $x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}$; $mc[x_{i_1}, \ldots, x_{i_k}]$ is the collection of documents whose vectors satisfy the monotone conjunction of $x_{i_1}, \ldots, x_{i_k}$; and $nc[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}]$ is the collection of documents whose vectors satisfy the non-monotone disjunction of $x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}$.

In this paper, we study collections of documents that are represented by disjunctions or conjunctions of k relevant variables. More precisely, we will design efficient algorithms to search for the following four concept classes of collections of documents given in Definition 3.2.

Definition 3.2. *Define*

$$MD(k) = \{md[x_{i_1}, \ldots, x_{i_k}] | 1 \leq k \leq n\},$$

$$ND(k) = \{nd[x_{i_1}, \ldots, x_{i_k}, x_{j_1}, \ldots, x_{j_m}] | l + m = k \text{ and } 1 \leq k \leq n\},$$

$$MC(k) = \{mc[x_{i_1}, \ldots, x_{i_k}] | 1 \leq k \leq n\},$$

$$NC(k) = \{nc[x_{i_1}, \ldots, x_{i_k}, x_{j_1}, \ldots, x_{j_m}] | l + m = k \text{ and } 1 \leq k \leq n\}.$$

Searching the Web with Membership Queries

In this section we study the problem of searching for documents with membership queries. We first design an algorithm to search for $md[x_{i_1}, \ldots, x_{i_k}]$, and then use the algorithm as a main module to search for $nd[x_{i_1}, \ldots, x_{i_k}]$, $mc[x_{i_1}, \ldots, x_{i_k}]$, and $nc[x_{i_1}, \ldots, x_{i_k}]$.

Because $md[x_{i_1}, \ldots, x_{i_k}]$ is logically equivalent to the monotone disjunction the k relevant variables,

$$x_{i_1} \vee \cdots \vee x_{i_k}, \qquad (*)$$

to search for $md[x_{i_1}, \ldots, x_{i_k}]$ we only need to identify the above monotone disjunction. Our idea is to use binary search to identify the relevant variables. Note that for any document d, $d \in md[x_{i_1}, \ldots, x_{i_k}]$ if and only if $v_d(x_1, \ldots, x_n)$ satisfies the disjunction given in (*). The algorithm is given as follows.

Algorithm 4.1.

Initially, let $S = \{1, 2, \ldots, n\}$, $t = 1$ and $R = \emptyset$.
while ($t \leq k$ and S is not empty) do
  Set $I_t = S$
  while I_t is not empty do
    Partition I_t into I_t^1 and I_t^2 such that $|I_t^1| = \lfloor \frac{|I_t|}{2} \rfloor$ and $|I_t^2| = \lceil \frac{|I_t|}{2} \rceil$.
    The search engine chooses a vector v such that for all $i \in I_t^1$, the i-th bit of v is 1 and, for any $i \notin I_t^1$ the i-th bit of v is zero.
    Ask a membership query for v.
    If the user responds with "No", then set $I_t = I_t^2$.
    If the user responds with "Yes", if $|I_t^1| = 1$ then let $R = R \cup I_t^1$ and $I_t = \emptyset$, otherwise let $I_t = I_t^1$.
  Set $t = t + 1$ and $S = S - R$.
If $R = \emptyset$, then the search engine responds to the user that "No documents found", otherwise the engine displays all the documents whose vectors satisfy the disjunction of the variables in R.

Theorem 4.2. *Algorithm 4.1 identifies any collection* $md[x_{i_1}, \ldots, x_{i_k}] \in MD(k)$ *of documents with at most* $k \log_2 n$ *membership queries.*

Proof. The number of membership queries needed by Algorithm 4.1 is at most $\sum_{t=1}^{k} \log_2 |I_t| \leq k \log_2 n$. □

Now, we design an algorithm to search for any collection $nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}]$ of documents whose vector satisfy the non-monotone disjunction

$$v(x_1, \ldots, x_n) = x_{i_1} \vee \cdots \vee x_{i_l} \vee \bar{x}_{j_1} \vee \cdots \vee \bar{x}_{i_m}, l + m = k. \ (**)$$

Because $nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}]$ is logically equivalent to (**), in order to identify it we only need to find (**). Given any vector $y \in \{0, 1\}^n$ such that y does not satisfy (**), then we have $y_{i_1} = \cdots = y_{i_l} = 0$ and $y_{j_1} = \cdots = y_{j_m} = 1$. Let $v^*(x_1, x_2, \ldots, x_n) = v(x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$, then

$$v^*(x_1, x_2, \ldots, x_n) = x_{i_1} \vee \cdots \vee x_{i_l} \vee x_{j_1} \vee \cdots \vee x_{i_m}.$$

Because

$$v(x_1, x_2, \ldots, x_n) = v^*(x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n),$$

in order to find $v(x_1, x_2, \ldots, x_n)$ we only need to find $v^*(x_1, x_2, \ldots, x_n)$. Thus, if a vector y that does not satisfy $v(x_1, x_2, \ldots, x_n)$ is given, by Theorem 4.2, at most $k \log_2 n$ membership queries are required to identify $v(x_1, x_2, \ldots, x_n)$. Note that such a vector y can be obtained by one equivalence query. The following corollary follows from the above analysis.

Corollary 4.3. *The collection* $nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}] \in ND(k)$ *of documents can be identified with exactly one equivalence query and at most* $k \log_2 n$ *membership queries, where* $l + m = k$.

Now, we consider how to search for the collection $mc[x_{i_1}, \ldots, x_{i_k}]$

of documents. Note that $mc[x_{i_1},\ldots,x_{i_k}]$ is logically equivalent to $u(x_1,\ldots,x_n)$ given below,

$$u(x_1,\ldots,x_n) = x_{i_1} \wedge \cdots \wedge x_{i_k}.$$

Since

$$\bar{u}(x_1,\ldots,x_n) = \bar{x}_{i_1} \vee \cdots \vee \bar{x}_{i_k},$$

let

$$u^*(x_1,\ldots,x_n) = \bar{u}(x_1+1,\ldots,x_n+1) = x_{i_1} \vee \cdots \vee x_{i_k},$$

then,

$$u(x_1,\ldots,x_n) = \overline{u^*}(x_1+1,\ldots,x_n+1).$$

Hence, in order to learn $u(x_1,\ldots,x_n)$, we only need to learn $u^*(x_1,\ldots,x_n)$. Thus, by Theorem 4.2 we have

Corollary 4.4. *There is an algorithm for identifying any collection $mc[x_{i_1},\ldots,x_{i_k}] \in MC(k)$ of documents with at most $k\log_2 n$ membership queries.*

To search for the collection $nc[x_{i_1},\ldots,x_{i_l},x_{j_1},\ldots,x_{j_m}]$ of documents, where $l+m=k$, we only need to learn

$$w(x_1,\ldots,x_n) = x_{i_1} \wedge \cdots \wedge x_{i_l} \wedge \bar{x}_{j_1} \wedge \cdots \wedge \bar{x}_{i_m},$$

because $nc[x_{i_1},\ldots,x_{i_l},x_{j_1},\ldots,x_{j_m}]$ is logically equivalent to $w(x_1,\ldots,x_n)$. Note that

$$\bar{w}(x_1,\ldots,x_n) = \bar{x}_{i_1} \vee \cdots \vee \bar{x}_{i_l} \vee x_{j_1} \vee \cdots \vee x_{i_m}.$$

For any y satisfying $w(x_1,\ldots,x_n)$, let $w^*(x_1,\ldots,x_n) = \bar{w}(x_1+y_1,\ldots,x_n+y_n)$, then

$$w^*(x_1,\ldots,x_n) = x_{i_1} \vee \cdots \vee x_{i_l} \vee x_{j_1} \vee \cdots \vee x_{i_m}.$$

Moreover,

$$w(x_1,\ldots,x_n) = \overline{w^*}(x_1+y_1,\ldots,y_n+1).$$

Hence, in order to learn $u(x_1,\ldots,x_n)$, we only need to learn $w^*(x_1,\ldots,x_n)$. Because a vector y satisfying $w(x_1,\ldots,x_n)$ can be obtained by one equivalence query, the following corollary follows from Theorem 4.2.

Corollary 4.5. *$nc[x_{i_1},\ldots,x_{i_l},x_{j_1},\ldots,x_{j_m}] \in NC(k)$ can be identified with exactly one equivalence query and at most $k\log_2 n$ membership queries.*

Searching the Web with Equivalence Queries

In this section we use Littlestone's linear-threshold algorithm (Littlestone 1988) to search for documents represented by disjunctions (or conjunctions) of k relevant attributes. In our case, the algorithm maintains non-negative real-valued weights $w_1,\ldots,w_n$ for variables $x_1,\ldots,x_n$, respectively. It also maintains a real threshold θ. Initially, all weights have values 1, and $\theta = \frac{n}{ek}$, where e is the natural logarithm constant. The search engine uses the linear threshold algorithm as follow:

- The engine predicts any documents whose vectors satisfy $\sum_{i=1}^n w_i x_i > \theta$ as the desired documents and displays them to the user.
- The engine predicts all documents whose vectors do not satisfy the above inequality as the undesired documents.

When the user responds the search engine with a document which contradicts to the engine's prediction, the engine updates the weights in two ways:

- If the user responds the engine with an undesired document whose vector is x, then for any i-th bit x_i such that $x_i = 1$, sets the weight $w_i = 0$.
- If the user responds the engine with a desired document whose vector is x, then for any i-th bit x_i such that $x_i = 1$, sets the weight $w_i = ew_i$.

The first way is called an elimination, and the second is called a promotion. The following theorem is derived from Littlestone's Algorithm Winnow (Littlestone 1988), but we give a more careful analysis and thus obtain a better bound on the number of equivalence queries.

Theorem 5.1. *There is an algorithm which identifies any collection $md[x_1,\ldots,x_k] \in MD(k)$ of documents with at most $k1.885\log_2 \frac{n}{k} - k$ equivalence queries.*

Proof. Recall that $md[x_1,\ldots,x_k]$ is logically equivalent to

$$v(x_1,\ldots,x_n) = x_{i_1} \vee \cdots \vee x_{i_k}.$$

We first estimate the number of promotions required in the search process of the engine. When a promotion occurs, the users responds the engine with a desired document d. Because d is desired, its vector $v_d = (x_1,\ldots,x_n)$ satisfies $v(x_1,\ldots,x_n)$, i.e., there is at least one $j \in \{1,\ldots,k\}$ such that $x_{i_j} = 1$. and for such a j the weight w_{i_j} is promoted to ew_{i_j}. On the other hand, for any $j \in \{1,\ldots,k\}$, the weight w_{i_j} can be promoted at most $\log_e \theta$ times, because if w_{i_j} has been promoted $\log_e \theta$ times, the weight w_{i_j} becomes $e^{\log_e \theta} = \theta$, thus it can not be promoted further. Hence, the total number of promotions is at most $k\log_e \theta$.

We now estimate the total number of eliminations required during the searching process of the engine. When an elimination occurs, the user responds the engine with an undesired document whose vector x satisfy $\sum_{i=1}^n w_i x_i > \theta$. Thus, after an elimination the total weight $\sum_{i=1}^n w_i$ is decreased by at least θ. When a promotion occurs, the user responds the engine with a desired document whose vector x satisfy $\sum_{i=1}^n w_i x_i \leq \theta$. Thus, after a promotion, the total weight $\sum_{i=1}^i w_i$ is increased by at most $(e-1)\theta$. Let u and v denote respectively the total numbers of promotions and eliminations occurred during the search process of the engine, then after those promotions and eliminations have occurred, the total weight $\sum_{i=1}^i w_i$

will be

$$\sum_{i=1}^{i} w_i \leq n + (e-1)\theta u - v\theta \leq n + (e-1)\theta k \log_e \theta - v\theta.$$

For any $j \in \{1, \ldots, k\}$, as we have analyzed before, the weight w_{i_j} can be promoted at most $\log_e \theta$ times. On the other hand, if w_{i_j} has been promoted less than $\log_e \theta$ times, then the user can still respond the engine with a desired document with a vector x such that the i_j-th bit of x is 1 but all other bits are 0. So, the weight w_{i_j} will be promoted at least one more time. This means that the weight w_{i_j} receives exactly $\log_e \theta$ many promotions. This fact implies that after the total u promotions have occurred, the total weight $\sum_{i=1}^{i} w_i$ will be at least $k\theta$. Thus, we have

$$k\theta \leq n + (e-1)\theta k \log_e \theta - v\theta.$$

It follows that

$$v \leq \frac{n}{\theta} + (e-1)k \log_e \theta - k.$$

Hence, the total number of promotions and eliminations is

$$u+v \leq \frac{n}{\theta} + (e-1)k \log_e \theta - k + k \log_e \theta = \frac{e}{\log_2 e} k \log_2 \frac{n}{k} - k.$$

Because $\theta = \frac{n}{ek}$, the right hand side of the above expression is bounded by

$$\frac{e}{\log_2 e} \log_2 \frac{n}{k} - k \leq 1.885k \log_2 \frac{n}{k} - k.$$

Hence, the total number of equivalence queries required by the search engine is at most $\leq 1.885k \log_2 \frac{n}{k} - k$. □

Now, we can use the above algorithm as a module to design new algorithms to search for collections of documents in $ND(k), MC(k)$, and $NC(k)$. With the similar analysis we have done for Corollary 4.3, 4.4, and 4.5, the following corollaries are derived from Theorem 5.1:

Corollary 5.2. *The collection* $nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}] \in ND(k)$ *of documents can be identified with at most* $1.885k \log_2 \frac{n}{k} - k + 1$ *equivalence queries, where* $l + m = k$.

Corollary 5.3. *The collection* $mc[x_{i_1}, \ldots, x_{i_k}] \in MC(k)$ *of documents can be identified with at most* $1.885k \log_2 \frac{n}{k} - k$ *equivalence queries.*

Corollary 5.4. *The collection* $nd[x_{i_1}, \ldots, x_{i_l}, x_{j_1}, \ldots, x_{j_m}] \in ND(k)$ *of documents can be identified with at most* $1.885k \log_2 \frac{n}{k} - k + 1$ *equivalence queries, where* $l + m = k$.

Discussions and Open Problems

According to (Lawrence & Giles 1998) there are at least 320 million indexable web documents over the Internet. The growth of the size of the Internet is rapid. But if we assume that size is doubled every year, then within the next 10 years, the number of indexable web documents will be about 2^{39}, since $2^{29} = 536,870,912 > 320,000,000$. This means the vector space that is capable of representing indexable web documents over the Internet within the next 10 years can be $\{0,1\}^{39}$. Note that $\log_2 39 < 5$. If on average a collection of desired web documents can be determined by disjunctions (or conjunctions) of 5 attributes, then our algorithms show that the collection can be exactly identified with at most 30 membership queries, or with at most 25 equivalence queries. Of course the number of queries can be reduced if we relax the exact identification requirement to approximate identification. One method for the approximate identification is to combine our learning algorithms with the vector similarity ranking technique (Salton 1989).

Even though the analysis above tells us that within the next 10 years, the vector space $\{0,1\}^{39}$ is big enough for presenting indexable web documents over the Internet, it is not easy to find those 39 boolean attributes to build the vector space. Many efforts should be made along this direction in the future. One direction we are planning to study is how to accumulate attributes based on the category classification tree of Yahoo.

In practice, a vocabulary of keywords is used as a set of attributes to build a vector space (for example, (Bowman *et al.* 1994)). The size of a vocabulary could be very large, say, for example, 10,000 keywords. If such a large vocabulary is used, then our algorithms tell us that, to search for a collection of web documents represented by disjunctions (or conjunctions) of 5 relevant attributes, the search engine needs at most 60 membership queries, or at most 99 equivalence queries. If size of the vocabulary is 100,000, then our algorithms require at most 75 membership queries, or at most 137 equivalence queries, to search for a collection of web documents represented by disjunctions (or conjunctions) of 5 relevant attributes. Although those bounds on the numbers of queries are of theoretical interests and can be further reduced if the exact identification requirement is relaxed, they are too big for an average user to bear. Clearly, we need more efficient algorithms to build a search engine when a large vocabulary is used to build the vector space.

A vector space built with a vocabulary of 10,000 (or even 100,000) keywords will be sparse in the sense that the majority of the vectors in the space have no corresponding web documents, because we have estimated earlier that the size of the indexable web documents over the Internet may be about 2^{39} within the next 10 years. In the future we will study whether we can find some compression methods to improve the efficiency of a search engine. In reality, at each interaction between a user and a search engine, the user can provide the engine with repeated relevance feedback indicating that

some documents on a list are relevant and others are not desired. We do not know theoretically how much this kind of repeated relevance feedback can help to improve the efficiency of the search engine.

Another promising direction is to use a user's personal profile to project the whole vector space to a relatively small subspace and then restrict the search of that user within the subspace. This idea can be extended to a group of users such as a group of all computer science faculty members in a department. Using a user's personal profile on filtering information in web search has been studied by researchers (for examples, (Frelechoux & Kamba 1997; Gabber *et al.* 1997)). In a recent paper (Meng & Chen 1999), we built a personal profile using the word frequencies from the files that a user has on the user's computer. This profile is used to aid the filtering of search results returned by search engines. This localized personal profile approach improves the search accuracy without sacrificing user privacy. We are working on combining the personalization technique and the algorithms obtained in this paper to filter information in web search.

In our *COLT SEARCH* project (Chen, Meng, & Fowler 1999), we started to build an experimental search engine around the end of 1998. It is implemented on a Sun Ultra-1 workstation with storage of 27 Giga-bytes hard disk on a IBM R6000 workstation. The main goal of this project is to provide empirical result and experience for building an intelligent search engine with machine learning, especially on-line learning, algorithms. We implemented a web crawler to collect web documents. So far we have collected more than 3.9 million URL's. Using the list of 343 keyword we built an index database of 843,398 documents. We also implemented a personalized ranking mechanism developed in (Meng & Chen 1999). We designed two major algorithms for the engine: Extended Membership Query Search and Extended Equivalence Query Search. The first has been implemented. The second is being implemented. Our experiments support the theoretical bounds on the number of queries. However, they also revealed a number of problems. The most important one is that *the membership assumption* may not be practically true. There are many reasons behind this. One is that the indexing is a pure syntactic process, so it may not be able to represent the actual meaning of a document. The extended membership query (Chen, Meng, & Fowler 1999) helps to overcome the obstacle to some degree. But it heavily relies on the validity of the ranking mechanism. The time performance is another issue for further investigation. In many cases it takes more than 20 minutes to process a query. But we believe that the performance can be improved in the near future.

References

Angluin, D. 1987. Queries and concept learning. *Machine Learning* (2):319–432.

Armstrong, R.; Freitag, D.; Joachims, T.; and Mitchell, T. 1995. Webwatcher: A learning apprentice for the world wide web. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, 6–12.

Bowman, C.; Danzig, P.; Hardy, D.; Manber, U.; and Schwartz, M. 1994. The harvest information discovery and access system. In *Proceedings of the Second International World Wide Web Conference*, 763–771.

Bshouty, N.; Chen, Z.; and Homer, S. 1994. On learning discretized geometric concepts. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 54–63.

Chen, Z., and Homer, S. 1996. The bounded injury priority method and the learnability of unions of rectangles. *Annals of Pure and Applied Logic* 77:143–168.

Chen, Z., and Maass, W. 1994. On-line learning of rectangles and unions of rectangles. *Machine Learning* 17:201–223.

Chen, Z.; Meng, X.; and Fowler, R. 1999. On building an intelligent www search engine with online learning algorithms. submitted for publication.

Chen, Z. 1995. *Computational Learning Algorithms for Geometric and Albegraic Objects.* Ph.D. Dissertation, Department of Computer Science, Boston University.

Etzioni, O. 1997. The world-wide web: Quagmire or gold mine? *Communication of ACM* 39:65–68.

Frelechoux, L., and Kamba, T. 1997. An architecture for support personalized web applications. In *Proceedings of the Sixth World Wide Web Conference.*

Gabber, E.; Gibbon, F.; Matias, Y.; and Mayer, A. 1997. How to make personalized web browsing simple, secure, and anonymous. In *Proceedings of Finanacial Cryptograph'97*, 24–28.

Lawrence, S., and Giles, C. L. 1998. Search the world wide web. *Science* 280:98–100.

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

Meng, X., and Chen, Z. 1999. Personlize web search using information on client's side. submitted for publication.

Perkowitz, M.; Doorenbos, R.; Etzioni, O.; and Weld, D. 1997. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems* (8):1–24.

Salton, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley.