December 2000

Personalization of Search Engine Services for Effective Retrieval and Knowledge Management

Weiguo Fan
University of Michigan

Michael Gordon
University of Michigan

Praveen Pathak
Purdue University

PERSONALIZATION OF SEARCH ENGINE SERVICES FOR EFFECTIVE RETRIEVAL AND KNOWLEDGE MANAGEMENT

Weiguo Fan
Michael D. Gordon
University of Michigan Business School
U.S.A.

Praveen Pathak
School of Management
Purdue University
U.S.A.

Abstract

The Internet and corporate intranets provide far more information than anybody can absorb. People use search engines to find the information they require. However, these systems tend to use only one fixed term weighting strategy regardless of the context to which it applies, posing serious performance problems when characteristics of different users, queries, and text collections are taken into consideration. In this paper, we argue that the term weighting strategy should be context specific, that is, different term weighting strategies should be applied to different contexts, and we propose a new systematic approach that can automatically generate term weighting strategies for different contexts based on genetic programming *(GP). The new proposed framework was tested on TREC data and the results are very promising.*

1. INTRODUCTION

The Internet has brought far more information than anybody can absorb. According to a recent study by NetSizer (NetSizer 2000), the number of web hosts has increased from 30,000,000 in January 1998, to 44,000,000 in January 1999, and to more than 70,000,000 in January 2000. More than 2,000,000 new hosts were added to the Internet in February 2000, according to this report. Similar Internet growth results were also reported by Internet Domain Service (IDS 2000). The number of web pages on the Internet was 320,000,000 million pages in December 1997, as reported by Lawrence and Giles (1997), 800,000,000 in February 1999 (Lawrence and Giles 1999), and more than 1,720,000,000 in March 2000 (Censorware 2000). The number of pages available on the Internet almost doubles every year.

Similar observations can also be made within an organization. As knowledge becomes a central productive and strategic asset, a company's success increasingly depends on its ability to gather, produce, maintain, and disseminate information and knowledge (Davenport and Laurence 1998). Knowledge management systems, which can help the organization systematically and actively manage and leverage the stores of knowledge, have found wide application within organizations. They put huge numbers of manuals, procedures, documentation, expertise knowledge, CD-ROMs, even e-mail archives, news resources, and technical reports on intranets and link them with other legacy information systems. The information/knowledge stored in these systems is also growing rapidly.

Such a vast amount of information serves as a huge information repository. However, it also makes finding relevant information on the Internet extremely difficult. Helping users find the information they require is the central task of most information retrieval (IR) systems or search engines.

The three key components of an IR system are (van Rijsbergen 1979):

1. Query representation—representing the user's information request
2. Document representation—representing the text collection
3. Matching Function or ranking function[1]—ranking the documents according to their relevance

In a typical IR process, a user enters the IR system with an information request. The request is expressed as a query and submitted to the IR system. The IR system extracts keywords from the query, matches the query against the document representations in its collection, and assigns a relevance status value (RSV) to each document. The whole document collection is then ranked in descending order based on RSV and only the top *n* documents are returned to the user.

Many models to index and represent documents as well as queries have been proposed in the IR literature. Among these, the vector space model is the most widely used (van Rijsbergen 1979). We will focus our discussion solely on the vector space model because of its simplicity and intuitive geometric representation (Jones and Furnas 1987). Note that in the vector space model, each query and document is represented as a vector. The similarity between these two vectors as often measured by the Cosine function is commonly used to represent the RSV for that document. More details on the vector space model will be discussed in the next section.

The effectiveness of an IR system is normally measured by *recall* and *precisio*n. *Recall* is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the whole collection. *Precision* is defined as the number of the relevant documents retrieved to the total number of documents retrieved. Earlier evaluations of IR systems were based primariy on some standard small text collections, such as Cranfield, CISI, CACM, etc. (Salton 1989).

However, with the advent of the Internet, new questions have been posed. One is whether techniques developed for small text collections can be extended to much larger collections. Recent Text REtrieval (TREC) conferences (Harman 1993, 1996), the pioneering efforts to study this issue, have shown mixed results. A recent study by Gordon and Pathak (1999) indicated that the precision and recall of commonly used search engines such as AltaVista and Infoseek are usually very low. Users often have to sift through many web pages to find a small set of documents that satisfy their information needs.

The performance of an IR system can be affected by many factors: the ambiguity of query terms, unfamiliarity with system features, factors relating to the document representation (Lancaster and Warner 1993). Many approaches have been proposed to address these issues. For example, query expansion techniques based on a user's relevance feedback have been used to discover the user's real information need (Chen et al. 1998b; Harman 1992). Gordon (1988) presented a genetic algorithm approach for adaptive document representation.

One more very important factor, which is key to the performance of an IR system but often overlooked by most people, is the ranking/matching function. Most of our discussion will focus on this ranking function.

As mentioned earlier, the ranking function is used to order documents in terms of their predicted relevance to a particular query. However, people tend to have different criteria in terms of the interpretation of relevance for each retrieved document. A document relevant to person *A* may be totally irrelevant to person *B* for the same query, or vice versa. How to design a ranking function that is effective in most contexts is often related to the system designer's mental model or some other factors. In fact, it is very hard to design a ranking function that will be successful for every query, user, or document collection (which we will call contexts). One of the main reasons for performance differences among search engine results is due to the variety of implementations of ranking functions (Gordon and Pathak 1999).

On the other hand, we argue in favor of an approach that systematically adapts a ranking function and tailors it to the needs of different users. In particular, we will use *genetic programming* (GP) (Koza 1992), an inductive learning technique, for the adaptation purpose.

[1]We will use matching function, similarity function, and ranking function interchangeably.

This paper is organized as follows: In section 2, we review the related work on term weighting and discuss our basic idea of an adaptive ranking/matching function. In section 3, we briefly introduce the concepts of GA and GP and show example applications of GA/GP to IR. Section 4 gives the detailed description of our proposed systematic approach. We demonstrate the effectiveness of our approach in section 5 and discuss the implications, limitation and future research directions in section 6.

2. TERM WEIGHTING AND RELATED WORK

The objective of a ranking function is to match the documents in a text collection against a query and order them in descending order of their predicted relevance. In the vector space model, both documents and queries are represented by vectors. Suppose there are total t index terms in the whole collection, a given document D and query Q can be represented as follows:

$$D = (w_{d1}, w_{d2}, w_{d3}, ..., w_{dt})$$

$$Q = (w_{q1}, w_{q2}, w_{q3}, ..., w_{qt})$$

where w_{di}, w_{qi} (i = 1 to t) are term weights assigned to different terms for the document and query respectively. The similarity between a query and a document can be calculated by the widely used cosine measure (Salton and Buckley 1988):

$$Similarity\ (Q,\ D) = \frac{\sum_{i=1}^{t} w_{qi} \times w_{di}}{\sqrt{\sum_{i=1}^{t} (w_{qi})^2 \times \sum_{i=1}^{t} (w_{di})^2}} \tag{1}$$

Documents are then ordered by decreasing values of this measure. In the vector space model, these weights are commonly measured by their statistical properties or statistical features. For example, one of the most widely used statistical features in term weighting strategy is *term frequency* (TF), which measures how many times the term has appeared in the document or query. Another commonly used feature is the *inverse document frequency* (IDF), which can be calculated by log (*N/DF*), where *N* is the total number of documents in the text collection and *DF* is another feature that measures the number of documents in which the term has appeared in the document collection. More statistical features used in term weighting can be found in Salton (1989) and Salton and Buckley (1988).

A lot of research has been done in the past 10 years to study the effects of different term weighting strategies on the performance of an IR system or a search engine. We will briefly review some of the work done in this area.

The earliest and most widely-cited work is by Salton and Buckley (1988) using their experimental SMART system. Based on the cosine measure as shown in Equation (1), they explored different statistical features (TF, IDF, etc.) and their combinations (TF*IDF, etc.) and tested them on six small-scale standard text collections. These statistical features were manually selected and tested. The total number of combinations was more than 1,800. Their experimental results showed that different term weighting strategies using different statistical features will produce different performance for the same query in the same document collection. No strategy could be found that worked consistently well across all of the queries and text collections.

Singhal et al. (1996) studied term weighting strategies further using the SMART system. They found that the normalization factor, the denominator in Equation (1), also has an impact on the performance of the IR system. Based on TREC conference experiments (Harman 1993), they found that the relevance judgements tend to favor long documents over short documents since longer documents contain more information than short ones. By changing the normalization factor and favoring long documents in retrieval, the performance can be improved by more than 20%. Singhal et al. also developed a formula to help accommodate such an effect. The formula contains some parameters that need to be adjusted based on the distribution of the relevance documents.

Zobel and Moffat (1998) recently conducted an extensive study on ranking functions that are broader than Cosine-based functions and include similarity measures such as Inner product, Dice, Jaccard, etc. The primary purpose of their research was to identify which statistical features yield good retrieval behavior. They tested a wide range of these similarity measures using the TREC conference data. The results of their experiments showed that there is no single ranking function that is a clear winner. There is also little overlap between the successful ranking functions in all of the test cases.

Rorvig (1999) studied the impact of ranking/similarity functions on visual information retrieval. In visual IR, not only the similarity between query and document, but also the relationships among documents need to be visualized. Rorvig used multi-dimensional scaling to visualize document similarities using five different similarity functions. Experimental results showed that among the five similarity functions, Cosine measure produces the tightest grouping of the data with the least dispersion of relevant documents. This may be partly due to the term weighting strategies incorporated in the Cosine measure as suggested by Rorvig. His experiment supports our work on the adaptation of similarity functions based on the Cosine measure.

Most of the work mentioned above is based on the vector space model. A key finding in all of these studies is that a single ranking function cannot work well for all contexts.

However, the term weighting strategies reported in these experiments are based primarily on IR experts' expertise, observations, and heuristics and are adjusted manually by these experts since they never know the optimal weighting strategy in advance. However, such a manual process is very tedious and expensive. No matter how the parameters of the ranking function are adjusted, the final performance of the ranking function for different contexts may still be questionable.

So one can easily ask, *Is there any automated system that can learn the "best" term weighting strategies for different contexts*? These contexts can be queries, users, or document collections. The answer is yes. The technique proposed in this paper can be used to replace the experts to automatically identify a better term weighting strategy.

We propose a new genetic programming approach to automatically adapt the term weighting strategies for ranking purposes. Like all other learning systems, feedback data are required to train the GP system. We use relevance feedback (RF) data gathered from the user for a particular search query as the training data. In fact, RF techniques have been widely used in IR to perform query expansions and adaptation of document representations (Chen et al. 1998b; Gordon 1988; Harman 1992). The idea of RF is that when a search engine retrieves a set of documents, the user will tell the system whether these documents are relevant to his or her information needs. These data are captured and used for later learning and adaptation.

Based on the user's relevance feedback for sample documents, our proposed approach can learn and approximate the user's underlying ranking function and apply it to unseen documents. We believe the proposed approach is very suitable for learning the needs of different users under different contexts, and thus can be modified and extended to address a wide range of information retrieval and mining problems. It is especially helpful for the design of personalized querying and retrieval system, and personalized information filtering and delivery.

3. GENETIC ALGORITHMS/PROGRAMMING AND THEIR APPLICATIONS TO IR

Genetic algorithms (Goldberg 1989; Holland 1992) and genetic programming (Koza 1992) are problem-solving systems based on the principles of evolution and heredity. Genetic programming (GP) is an extension of the genetic algorithm (GA). In a word, GA and GP are essentially iterative, parallel global search algorithms that are widely used to solve optimization and approximation problems. We next introduce GAs to help explain GPs.

In a GA, a population of individuals is maintained at each generation. Each individual represents a possible solution to the problem at hand and is assigned a fitness value to guide the search. Individuals with higher fitness values are selected and undergo genetic transformation by genetic operators. Two genetic operators are widely used: *crossover* and *mutation.* The crossover operator randomly selects two individuals as parents and exchanges parts of their construction to form two new individuals. The mutation operator merely randomly selects one individual from the parent population and changes its internal representation and puts it in the child population. Both of the crossover and mutation rates should be carefully adjusted to improve search performance. The newly generated child population becomes the parent population for the next generation and undergoes the same process again and again until a stopping criterion has been satisfied.

The difference between GA and GP is the internal representation or data structure for those individuals. In GA, each individual is commonly represented by a fixed-length bit strings, like (1101110…), or a fixed sequence of real numbers (1.2, 2.4, 4, …). In GP, more complex data structures, e.g., trees, linked list, and stacks, are used (Langton 1998). The length of the data structure is not fixed, but can be constrained by implementation. GA is normally used to solve some hard parameter optimization problems, while GP is widely used to approximate complex nonlinear functional relationships (Koza 1992). A detailed explanation of GP implementation using a tree structure will be presented in the next section.

The application of GA/GP to intelligent information retrieval has attracted a lot of research attention recently.

The earliest pioneering work in this area was done by Gordon (1988). He presented a GA-based approach to document indexing. Competing document descriptions (keywords) are associated with a document and altered over time by genetic crossover and mutation operators. He used bit strings in his GA design with each bit in the string representing a keyword. The whole string represents the list of key words within one document. A collection of documents initially judged relevant by a user was used as the initial population. Based on the Jaccard's score matching function (fitness measure), the initial population was evolved under the crossover and mutation operators generation by generation and eventually converged to the "optimal" (improved) population, which corresponds to a set of keywords that best describe the documents. Following a similar approach, Gordon (1991) studied the problem of document clustering using genetic algorithms. His experimental results showed that after genetically redescribing the subject description of documents, descriptions of documents found co-relevant to a set of queries will bunch together, forming a cluster. Re-description improved the relative density of co-relevant documents by 39.74% after 20 generations and 56.61% after 40 generations. The same problem was also studied by Raghavan and Agarwal (1987).

Petry et al. (1993) applied genetic algorithms to a weighted information retrieval system. In their research, a weighted Boolean query was modified in order to improve recall and precision. They found that the form of the fitness function has a significant impact on the performance. Yang and his colleagues (Yang and Korfhage 1993; Yang et al. 1993) developed adaptive retrieval methods by changing the weights assigned to the query terms based on feedback from the user. They reported the effect of adopting genetic algorithms in large-scale databases, the impact of genetic operators, and the GA's parallel search capability.

Chen et al. (1998a) studied the query by example problem to learn the user's information need by experimenting with several machine learning techniques. Users submitted a sample of documents representing their interests, then various learning systems tried to learn the keywords or concepts that best represented the users' information quest. Their results indicated that a learning system based on GA did consistently better than other approaches. They continued the application of GA and designed Internet spiders based on best-first search and GA search (Chen et al. 1998b). Based on the initial list of home pages provided by the user, the spider uses these two search methods to browse through the Internet and report findings that match the user's interest. They found that a spider based on GA generates more new relevant documents than the best-first search spider. A similar Internet agent/spider system based on GA was also developed by Martin-Bautista et al. (1999). They represent the query keywords extracted from the homepages as the individuals of the GA population and try to learn the user's interest based on the user feedback by modifying the keywords within each individual. The agent then searches the web and gives a ranked list of the documents to the user. These documents are ranked based on the score by aggregating the similarity between the document and each of the individuals in the final population.

Pathak et al. (2000) applied the GA to adaptively change the matching function during a search session. In their research, a linear combination of several well-known traditional matching functions, such as Dice, Jaccard, Cosine, etc., was used as the overall matching function. The weight for each of the matching functions is represented as an individual in the population. A set of documents of known relevance is used to train the GA to find the "best" weights for the final matching function.

The work reported in this paper is another novel approach to the adaptation of matching functions.

4. AUTOMATIC TERM WEIGHTING BY GENETIC PROGRAMMING

In this paper, we propose a GP-based approach to the study of the problem of automatic term weighting. A tree data structure is used in our GP implementation. We found that GP is extremely suitable for the term weighting task as it can automatically combine the various clues or features to approximate the user's information need.

An example of representing one term weighting formula using a tree structure is shown in Figure 1. A typical term weighting formula, as shown on the right side of the arrow, is internally represented using a tree data structure, as shown on the left side of the arrow. The tree-based representation allows for ease of parsing and implementation. We apply the term weighting formula to each term within a document to calculate the document's similarity to the query.

The detailed description of our genetic programming based approach follows.

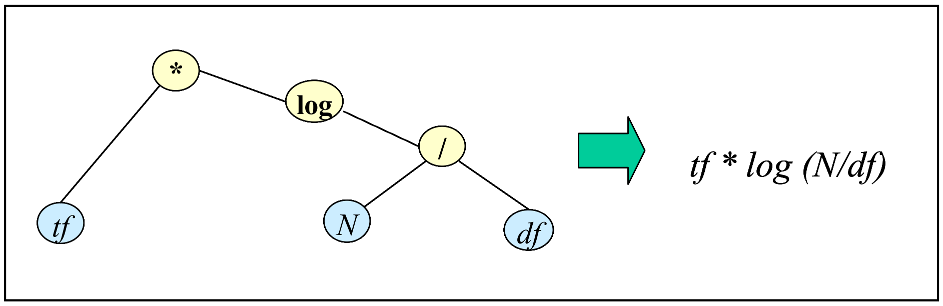

Figure 1. A Sample Term Weighting Formula Generated by the GP System

4.1 Terminals

In GP, the terminals are the leaf nodes of the tree data structure (as shown in Figure 1). They are essentially many statistical features used in term weighting. The terminals described in Table 1 are used in our GP implementation.

Table 1. The Terminals Used in the GP System

Tf	Same as TF: how many times the term appeared in a document
tf_max	The maximum tf for a document
tf_avg	The average tf for a document
tf_doc_max	The maximum tf in the whole document collection
Df	Same a DF: the number of unique documents in which the term appeared
df_max	The maximum df for the whole collection
N	The total number of documents in the whole text collection
Length	The length of a document
Length_avg	The average length of all documents in the whole collection
R	The real constant number randomly generated by the GP system
N	The number of unique terms in a document

4.2 Functions

Functions are the operations that will be applied to the terminals and to sub-trees. The following functions were used in our implementation:

$$+, -, \times, /, \log$$

4.3 Initial Population Generation

In GP, the population is a set of individuals that will be evolved by genetic operators. They represent the candidate term weighting formula for a particular context. Since the query terms in the query vector normally appear only once (tf = 1) and they don't provide more information in terms of the relevance, we use the TF as the default term weighting strategy for the query vector. By varying the term weighting strategies for the document vector and computing its Cosine value with the query vector, different matching/ranking functions can be generated.

The initial population normally has some constraints on the depth of the tree to allow a good sampling of terminals and functions. For example, our initial population is constrained to be of depth 2 to 4. We use the Ramped Half-and-Half method (Koza 1992) to generate the initial population. That is, half of the population is generated using the *grow* method that randomly selects the nodes of the tree from the function and terminal set for each depth. If it selects a terminal node in an intermediate depth, it will stop selecting along that branch of the tree, otherwise, it will keep selecting until it selects a terminal node or the depth limit is reached. The other half of the population is generated using the *full* method that selects only a function node for intermediate depths and a terminal node for the last depth. The Ramped Half-and-Half method has been found to generate a good variety of samples (Koza 1992).

4.4 The GP Operators

The GP operators used in our implementation are *reproduction* and *crossover.*

For reproduction, we first rank the whole population in descending order based on their fitness values. Depending on the system setting on the reproduction rate (*rate_r*) and the population size (*P*), the top (*P* * *rate_r*) individuals are copied directly to the next generation without any modification or transformation. The intuition behind this is that the best individuals may contain some good clues that are useful for improvement of fitness values and, by copying them into the next generation, they have more chances to be selected for crossover in the next generation. The balance of the individuals is generated by crossover.

For crossover, we first randomly select six individuals from the current population.[2] Two out of the six that have higher fitness values are chosen as the parents for the crossover operation. The other four will be passed over. For each parent, a crossover point is randomly selected. The parents then exchange the sub-trees below their crossover points and copy themselves to the next generation as two newly generated individuals.

4.5 Fitness Functions

The fitness function measures how good the individual is at the current generation. In our implementation, we use the precision of the top DCV[3] documents as the fitness value for each individual. The calculation of the fitness value for an individual in a particular generation is as follows:

> For each document in the entire text collection, assign a retrieval status value (RSV) by the tree representing the current individual. Then these documents will be ranked in descending order of the RSV and the top DCV documents will be selected. Since these documents have been judged relevant *a priori*, the precision of the top DCV documents will be the total number of relevant documents in the top DCV documents divided by the DCV.

Of course, other fitness functions can be used (Yang and Korfhage 1993; Yang et al. 1993). We found this particular fitness function to be simple and easy to understand. It is also proved to work well in our experiments.

[2]The selection method we used here is called tournament selection (Koza 1992). The tournament size can bechanged via the system settings.

[3]DCV, means the document cutoff value. It is a system parameter for the GP system.

4.6 Termination Criteria

The GP will evolve a population of individuals generation by generation using crossover and reproduction. Since we don't know the best solution and the whole procedure is a discovery process, the GP system will be stopped after a predefined maximum number of generations is reached. We chose 15 as the maximum number of generations in our implementation since, in our trial runs, we found that GP will not improve dramatically after 15 generations.

4.7 The Whole Discovery Process

To summarize, the whole learning process based on GP is:

1. Generate an initial population of random compositions of the terminals and functions.

2. Perform the following sub-steps iteratively on the training documents provided by the user until the final termination criterion has been satisfied:
   a. Execute every individual's program and calculate its fitness value based on the relevance feedback from the users.
   b. Create a new population by applying to the selected individuals
      - Reproduction
      - Crossover

3. Report the top 10 term weighting strategies and their corresponding performance measures on the testing documents.

5. EXPERIMENT

Two separate experiments were conducted to test the viability of the proposed approach. The first experiment was on a small-scale text collection, Cranfield, which contains 1,400 documents. A total of 225 queries were judged for relevance against these 1,400 documents. The second experiment was on a much larger text collection, the Federal Register (FR) text collection from TREC 4. The total number of documents in this collection is 55,554. We ran all 50 queries on the above TREC data. Due to page limitations, we will only report the results on TREC data.

5.1 Experimental Setting

TREC is the primary conference focusing on the applicability of traditional as well as new IR techniques on large-scale text collections (Harman 1993). Many information retrieval systems and commercial engines have used TREC data as the test bed to validate and evaluate their performance. Compared to the other conference data, TREC 4 is the only one that is focused on short queries. Since many of the queries submitted to search engines and online retrieval systems are very short, mostly of two to nine keywords (Gordon and Pathak 1999), we feel that the study of searching on the TREC 4 data will give us more insights on the design and evaluation of a real search engine.

As argued earlier, term weighting strategy should be context specific. Here we assume that each query defines a separate context, so different term weighting strategies are generated for each query.

There are a total of 50 queries used in TREC 4. Each of the 50 queries is on very different topic. The 55,554 documents in the FR collection have already been judged for relevance by experts. The number of terms contained in the 50 queries varies from four to 18. The total number of relevant documents in the whole FR collection for the 50 queries varies from four to 407. The wide variety of queries and the documents relevant to them provide a robust test environment for our study.

We sampled the 55,554 documents by including a small number of relevant documents for each query. By doing so, we generated a single training document set that contains 2,200 documents. This document set serves as the common training data set for every query.

Our experience with the Cranfield text collection suggested that we use large population size and different random seeds on the TREC data. Our experiment is set up as follows.

Training:

Training data	2,200 documents sampled from 55,554 documents with bias on the relevant documents
DCV	50
Population size	200
Random seeds	5
Crossover rate	0.9
Reproduction rate	0.1
Generations	15

Testing:

Test data	The whole 55,554 documents

Preliminary experiments showed that there might have been some over-fitting of the data (discussed below). The experiment proceeds as follows: for each query, we train the GP system using the 2,200 documents. At the end of each generation of the GP training process, the 10 best trees for that generation are tested on the 55,554 test documents and their performance is recorded for later analysis. So 15 * 10 (= 150) performance results are saved. The best performing tree is then selected from these 150 trees for each query. This procedure allowed us to examine the possible effects of over-fitting.

5.2 Results and Discussion

The best results for our GP system are summarized in Table 3. For a benchmark comparison, we also applied the TF*IDF term weighting strategy on the TREC 4 data and summarized the results in Table 4. The comparison between GP and TF*IDF on three performance measures is summarized in Table 5. The statistical tests t-test and MANOVA (Hull 1993) on the difference between GP and TF*IDF for all the performance measures are shown in Table 6.

The performance measures used in Tables 3, 4, 5, and 6 are defined in Table 2.

Table 2. Definition of Performance Measures

Name	Meaning
Q_No.	Query number.
Run No.	The index number of the five different runs.
Gen No.	Generation number in which the best result on test data appears.
Tree No.	The number of the tree (out of 10) that generates the best result on the testing data.
P_avg	Whenever a new relevant document is retrieved, the precision is added up to the sum. P_avg is the sum divided by the no. of relevant documents retrieved.
R_avg	Similar to P_avg, but for recall.
P_dcv_avg	The average precision by averaging precision at DCV= 50, 100, 150, 200, ..., 1,000.
R_dcv_avg	The average recall by averaging recall at DCV= 50, 100, 150, 200, ..., ,1000.
R_P	The precision when T_Rel documents are retrieved.
T_R_P	The number of relevant documents when T_Rel documents are retrieved.
T_Recall	The recall of the top 1,000 documents retrieved.
T_Rel_Ret	The number of relevant documents in the top 1,000 documents retrieved.
T_Rel	The total number of relevant documents in the whole text collection.

Table 3. Summary of GP System[a]

Q_No.	Run No.	Gen No.	Tree No.	P_avg	R_avg	P_dcv_avg	R_dcv_avg	R_P	T_R_P	T_Rel_ Ret	T_Rel	T_Recall
1	4	8	6	0.26	0.40	0.08	0.63	0.29	14	38	49	0.78
2	3	12	4	0.19	0.42	0.04	0.69	0.17	3	14	18	0.78
3	2	14	5	0.07	0.46	0.03	0.74	0.07	1	12	14	0.86
4	3	10	2	0.33	0.52	0.07	0.88	0.38	10	26	26	1.00
5	2	9	9	0.27	0.30	0.03	0.42	0.27	6	12	22	0.55
6	4	8	9	0.22	0.19	0.10	0.24	0.16	24	59	154	0.38
7	2	11	3	0.32	0.52	0.07	0.92	0.35	9	26	26	1.00
8	4	13	0	0.31	0.40	0.20	0.62	0.32	41	103	130	0.79
9	3	10	4	0.09	0.44	0.05	0.63	0.15	6	33	39	0.85
10	2	7	0	0.38	0.50	0.06	0.91	0.36	8	21	22	0.95
11	4	4	4	0.41	0.37	0.07	0.60	0.37	14	27	38	0.71
12	4	15	7	0.64	0.43	0.31	0.70	0.52	84	137	162	0.85
13	1	9	0	0.55	0.49	0.10	0.90	0.49	17	33	35	0.94
14	4	6	7	0.13	0.27	0.12	0.35	0.15	25	86	162	0.53
15	4	7	5	0.14	0.60	0.01	0.88	0.20	1	5	5	1.00
16	3	9	3	0.19	0.48	0.05	0.77	0.27	6	20	22	0.91
17	1	8	2	0.33	0.37	0.19	0.56	0.34	48	105	143	0.73
18	3	4	4	0.21	0.45	0.07	0.68	0.30	11	32	37	0.86
19	5	8	3	0.34	0.48	0.06	0.70	0.30	10	31	33	0.94
20	4	4	3	0.52	0.28	0.20	0.47	0.41	60	80	146	0.55
21	2	8	5	0.22	0.63	0.01	0.86	0.25	1	4	4	1.00
22	3	11	9	0.35	0.41	0.05	0.72	0.30	7	18	23	0.78
23	4	7	8	0.26	0.21	0.22	0.26	0.23	93	170	407	0.42
24	4	2	5	0.43	0.35	0.03	0.54	0.41	7	11	17	0.65
25	5	15	0	0.21	0.46	0.04	0.68	0.17	4	21	24	0.88
26	3	12	3	0.27	0.38	0.07	0.64	0.26	10	28	38	0.74
27	1	14	4	0.40	0.44	0.07	0.69	0.40	14	30	35	0.86
28	5	12	0	0.19	0.36	0.08	0.53	0.14	9	47	66	0.71
29	1	5	0	0.40	0.25	0.03	0.39	0.19	6	15	32	0.47
30	5	4	6	0.32	0.42	0.11	0.72	0.31	17	44	54	0.81
31	5	1	1	0.17	0.38	0.05	0.60	0.21	6	20	28	0.71
32	5	11	9	0.22	0.23	0.08	0.36	0.22	17	34	76	0.45
33	2	15	0	0.06	0.19	0.04	0.26	0.09	6	26	70	0.37
34	1	5	9	0.41	0.39	0.22	0.54	0.31	50	127	163	0.78
35	1	15	9	0.12	0.43	0.01	0.54	0.29	2	5	7	0.71
36	5	13	1	0.11	0.38	0.06	0.55	0.14	5	27	37	0.73
37	1	8	3	0.17	0.34	0.02	0.41	0.18	4	14	22	0.64
38	3	12	1	0.25	0.42	0.12	0.67	0.31	20	53	65	0.82
39	3	5	0	0.13	0.30	0.03	0.44	0.20	4	11	20	0.55
40	1	9	9	0.22	0.24	0.13	0.35	0.13	20	76	160	0.48
41	2	14	3	0.47	0.43	0.08	0.70	0.49	17	29	35	0.83
42	2	13	8	0.22	0.33	0.03	0.40	0.21	5	15	24	0.63
43	3	12	7	0.15	0.38	0.02	0.51	0.18	3	12	17	0.71
44	2	11	0	0.30	0.49	0.08	0.80	0.35	12	32	34	0.94
45	2	3	8	0.31	0.25	0.16	0.40	0.30	43	71	145	0.49
46	4	14	8	0.09	0.47	0.06	0.69	0.14	5	32	35	0.91
47	4	4	0	0.19	0.30	0.11	0.44	0.21	21	57	98	0.58
48	4	12	0	0.48	0.22	0.26	0.33	0.29	87	133	305	0.44
49	5	5	3	0.27	0.44	0.05	0.75	0.44	8	15	18	0.83
50	1	13	0	0.44	0.48	0.06	0.86	0.43	10	21	23	0.91
Sum.				**13.71**	**19.34**	**4.28**	**29.90**	**13.63**	**911**	**2098**	**3365**	**37**
Avg.				**0.27**	**0.39**	**0.09**	**0.60**	**0.27**	**18**	**41.96**	**67**	**0.74**
											Mac_R:	**0.62**

[a]This table shows how the GP fared for 50 queries. Columns 5 through 11 and column 13 indicate different performance measures. Note that column 5 corresponds to precision, the performance measure the simulation was run to improve. Please see Table 2 for definitions of the performance measures.

Table 4. Summary of TF*IDF Term Weighting Strategy on TREC 4 Data[a]

Q_No.	P_avg	R_avg	P_dcv_avg	R_dcv_avg	R_P	T_R_P	T_Rel_ret	T_Rel	T_Recall
1	0.26	0.24	0.06	0.41	0.18	9	23	49	0.47
2	0.00	0.08	0.00	0.01	0.00	0	2	18	0.11
3	0.02	0.32	0.01	0.42	0.00	0	8	14	0.57
4	0.08	0.19	0.02	0.29	0.08	2	9	26	0.35
5	0.00	0.00	0.00	0.00	0.00	0	0	22	0.00
6	0.14	0.08	0.03	0.09	0.03	5	23	154	0.15
7	0.00	0.00	0.00	0.00	0.00	0	0	26	0.00
8	0.05	0.01	0.00	0.01	0.01	1	1	130	0.01
9	0.04	0.24	0.03	0.32	0.05	2	18	39	0.46
10	0.01	0.09	0.00	0.09	0.00	0	3	22	0.14
11	0.00	0.04	0.00	0.01	0.00	0	2	38	0.05
12	0.15	0.20	0.10	0.27	0.14	22	63	162	0.39
13	0.00	0.03	0.00	0.02	0.00	0	1	35	0.03
14	0.00	0.01	0.00	0.00	0.00	0	2	162	0.01
15	0.00	0.30	0.00	0.23	0.00	0	2	5	0.40
16	0.00	0.00	0.00	0.00	0.00	0	0	22	0.00
17	0.01	0.01	0.00	0.01	0.01	1	1	143	0.01
18	0.11	0.32	0.05	0.54	0.11	4	23	37	0.62
19	0.36	0.21	0.03	0.34	0.15	5	13	33	0.39
20	0.40	0.28	0.18	0.46	0.38	56	80	146	0.55
21	0.01	0.25	0.00	0.23	0.00	0	1	4	0.25
22	0.06	0.22	0.02	0.34	0.09	2	9	23	0.39
23	0.09	0.06	0.06	0.07	0.06	25	44	407	0.11
24	0.07	0.26	0.02	0.44	0.06	1	8	17	0.47
25	0.09	0.19	0.02	0.29	0.08	2	8	24	0.33
26	0.03	0.12	0.01	0.11	0.03	1	8	38	0.21
27	0.15	0.21	0.02	0.28	0.11	4	14	35	0.40
28	0.16	0.11	0.02	0.11	0.05	3	14	66	0.21
29	0.01	0.19	0.01	0.19	0.00	0	11	32	0.34
30	0.00	0.03	0.00	0.01	0.00	0	2	54	0.04
31	0.07	0.21	0.02	0.33	0.07	2	11	28	0.39
32	0.00	0.00	0.00	0.00	0.00	0	0	76	0.00
33	0.03	0.12	0.02	0.16	0.03	2	16	70	0.23
34	0.05	0.03	0.02	0.05	0.03	5	10	163	0.06
35	0.05	0.21	0.00	0.21	0.00	0	2	7	0.29
36	0.19	0.12	0.02	0.16	0.08	3	8	37	0.22
37	0.00	0.07	0.00	0.05	0.00	0	2	22	0.09
38	0.06	0.20	0.03	0.22	0.06	4	25	65	0.38
39	0.04	0.15	0.01	0.20	0.05	1	5	20	0.25
40	0.10	0.09	0.05	0.13	0.06	10	28	160	0.18
41	0.07	0.19	0.02	0.22	0.06	2	12	35	0.34
42	0.15	0.13	0.01	0.20	0.08	2	5	24	0.21
43	0.06	0.12	0.01	0.15	0.06	1	3	17	0.18
44	0.11	0.16	0.02	0.20	0.06	2	10	34	0.29
45	0.05	0.08	0.02	0.08	0.01	2	23	145	0.16
46	0.20	0.23	0.03	0.34	0.20	7	15	35	0.43
47	0.00	0.01	0.00	0.00	0.00	0	1	98	0.01
48	0.13	0.10	0.09	0.13	0.10	32	57	305	0.19
49	0.12	0.11	0.01	0.13	0.06	1	3	18	0.17
50	0.04	0.17	0.01	0.19	0.09	2	7	23	0.30
Sum	**3.80**	**6.80**	**1.15**	**8.70**	**2.72**	**223**	**636**	**3365**	**11.83**
Avg.	**0.08**	**0.14**	**0.02**	**0.17**	**0.05**				**0.24**
								Mac_R:	**0.19**

[a]This table shows how the TF*IDF fared for 50 queries. Columns 2 through 8 and column 10 indicate different performance measures. Note that column 2 corresponds to precision, the performance measure the simulation was run to improve. Please see Table 2 for definitions of the performance measures.

Table 5. Summary of GP and TF*IDF Comparison[a]

	P_avg			T_R_P			T_Rele_Ret.		
Q_No.	TFIDF	GP	Inc.(%)	TFIDF	GP	Inc.(%)	TFIDF	GP	Inc.(%)
1	0.26	0.26	-0.22%	9	14	55.56%	23	38	65.22%
2	0.00	0.19	11060.43%	0	3		2	14	600.00%
3	0.02	0.07	281.77%	0	1		8	12	50.00%
4	0.08	0.33	304.42%	2	10	400.00%	9	26	188.89%
5	0.00	0.27		0	6		0	12	
6	0.14	0.22	57.27%	5	24	379.99%	23	59	156.52%
7	0.00	0.32		0	9		0	26	
8	0.05	0.31	528.75%	1	41	4000.17%	1	103	10200.00%
9	0.04	0.09	92.47%	2	6	200.00%	18	33	83.33%
10	0.01	0.38	4473.70%	0	8		3	21	600.00%
11	0.00	0.41	23006.27%	0	14		2	27	1250.00%
12	0.15	0.64	322.62%	22	84	281.82%	63	137	117.46%
13	0.00	0.55	20884.26%	0	17		1	33	3200.00%
14	0.00	0.13	7035.53%	0	25		2	86	4200.00%
15	0.00	0.14	3823.40%	0	1		2	5	150.00%
16	0.00	0.19		0	6		0	20	
17	0.01	0.33	3345.20%	1	48	4700.00%	1	105	10400.00%
18	0.11	0.21	87.99%	4	11	175.00%	23	32	39.13%
19	0.36	0.34	-4.77%	5	10	100.00%	13	31	138.46%
20	0.40	0.52	30.57%	56	60	7.14%	80	80	0.00%
21	0.01	0.22	3112.10%	0	1		1	4	300.00%
22	0.06	0.35	448.15%	2	7	250.00%	9	18	100.00%
23	0.09	0.26	205.88%	25	93	272.00%	44	170	286.36%
24	0.07	0.43	564.56%	1	7	599.99%	8	11	37.50%
25	0.09	0.21	138.79%	2	4	100.00%	8	21	162.50%
26	0.03	0.27	847.23%	1	10	899.99%	8	28	250.00%
27	0.15	0.40	174.40%	4	14	250.00%	14	30	114.29%
28	0.16	0.19	15.50%	3	9	200.00%	14	47	235.71%
29	0.01	0.40	2876.32%	0	6		11	15	36.36%
30	0.00	0.32	17459.92%	0	17		2	44	2100.00%
31	0.07	0.17	128.98%	2	6	200.00%	11	20	81.82%
32	0.00	0.22		0	17		0	34	
33	0.03	0.06	90.06%	2	6	200.00%	16	26	62.50%
34	0.05	0.41	757.15%	5	50	899.99%	10	127	1170.00%
35	0.05	0.12	155.83%	0	2		2	5	150.00%
36	0.19	0.11	-39.61%	3	5	66.67%	8	27	237.50%
37	0.00	0.17	5605.95%	0	4		2	14	600.00%
38	0.06	0.25	293.47%	4	20	400.00%	25	53	112.00%
39	0.04	0.13	213.15%	1	4	300.00%	5	11	120.00%
40	0.10	0.22	134.93%	10	20	100.00%	28	76	171.43%
41	0.07	0.47	543.89%	2	17	750.00%	12	29	141.67%
42	0.15	0.22	48.83%	2	5	150.00%	5	15	200.00%
43	0.06	0.15	147.07%	1	3	200.00%	3	12	300.00%
44	0.11	0.30	172.26%	2	12	499.99%	10	32	220.00%
45	0.05	0.31	568.57%	2	43	2050.02%	23	71	208.70%
46	0.20	0.09	-55.42%	7	5	-28.57%	15	32	113.33%
47	0.00	0.19	16244.11%	0	21		1	57	5600.00%
48	0.13	0.48	285.90%	32	87	171.88%	57	133	133.33%
49	0.12	0.27	129.31%	1	8	699.99%	3	15	400.00%
50	0.04	0.44	1114.24%	2	10	400.00%	7	21	200.00%
Sum	**3.80**	**13.71**	**127711.16%**	**223**	**911**	**19931.63%**	**636**	**2098**	**45284.02%**
Mic_Inc			**2776.33%**			**603.99%**			**984.44%**
Mac_Inc			**260.34%**			**308.52%**			**229.87%**

[a]This table compares GP with TF*IDF for three key performance measures. See Table 2 for definitions of these three performance measures.

Table 6. Statistical T-Test and MANOVA[a]

Performance Measure	T-Test	MANOVA	
	p-value	univariate F-value	p-value
P_avg	3.12E-13	77.57	4.67E-14
R_avg	1.80E-21	162.25	0
P_dcv_avg	7.98E-11	35.02	4.81E-8
R_dcv_avg	3.06E-19	162.11	0
R_P	3.96E-18	142.34	0
T_R_P	2.69E-07	16.05	1.20E-4
T_Rel_ Ret	1.33E-08	24.34	3.30E-6
T_Recall	1.08E-21	199.90	0
	Pillai Trace: 0.74922	F: 29.87605	P-value: 0

[a]This table shows the statistical tests on the performance comparisons between GP and TF*IDF based on eight performance measures. See Table 2 for definitions of the performance measures.

All of the results in Table 3, 4, and 5 are on the entire test data, that is, the entire FR document collection in TREC 4. These results are calculated by taking the top 1,000 retrieved documents and computing the relevance judgement for a particular query. Many of these measures are standard performance measures as used by the TREC conference for evaluation and comparison among different retrieval systems (Harman 1993).

If we look at the comparison results shown in Table 5, we see that we have found a huge improvement over the traditional TF*IDF approach. Note that the best term weighting strategies were selected based on the T_Rele_Ret performance measure, thus it may not represent the best of the other performance measures. That is also why we can find some dropping performance for some queries. The biggest differences on T_Rele_Ret were found on queries 8 and 12. Both are larger than 10,000%. In fact, we found that GP improves performance much more on queries that have a larger number of relevant documents. This may support the argument that GP may be a very powerful tool for the web search environment. We will test this hypothesis in our future research work. Another observation that can be made from Table 5 is that the overall performance improvement is very impressive, more than 200% over TF*IDF. The statistical test of the difference for all the performance measures, as shown in Table 6, strongly supports our conclusion that the performance gain made by using our approach over TF*IDF on the TREC 4 text collection is statistically significant.

As can be seen in Table 3, the best term weighting strategy for each individual query was obtained by different random seeds as reflected by the run number. This shows that random seeds have impact on the GP system's performance. One of the most interesting observations is that the matching function that performs best on the test data was not the one that performs best on the training data. This is the so-called "over-fitting" problem (Mitchell 1997), which is reflected in the generation number and tree number of Table 3. Trees within each generation are ranked by their performance on the training data, so tree 0 is supposed to be the best one for that generation. Tree 0 in the last generation is the one that has the best overall performance. If there are no over-fitting problems, that is, the trees with the best performance results on the training data should generalize well on the testing data, we would expect the values shown in the generation number and tree number columns to be 15 and 0, respectively. This is not the case in our experiment results.

In fact, over-fitting can happen in any learning and optimization system (Mitchell 1997). That is not the focus of our paper. Instead, we argue that by using learning tools such as GP we can identify much better term weighting formulas for different queries than those that are developed based solely on heuristics. We are more interested in the learning and discovery features and find them to be the most valuable. This argument can be supported by the results in Table 5.

6. IMPLICATIONS AND FUTURE WORK

We considered in this paper a novel approach to automatic term weighting by genetic programming. Since most search engines built on the vector space model use only a fixed ranking or term weighting strategy for all contexts (the so-called "one size fits all" limitation), this poses serious problems when they are used to satisfy a wide range of user information needs. Instead, by combining the common statistical clues or features used in the traditional term weighting strategies in an intelligent way using GP, we found that we can improve the retrieval performance quite dramatically. The results of the TREC conference data have demonstrated the advantages of our GP approach.

We believe our work has the following important implications:

- Adaptive search engines. Future search engines should be adaptive, interactive and intelligent (Martin-Bautista et al. 1999). They should be able to change their ranking strategies by inferring the users' preference. Our work serves as a pilot in this area and provides a solid foundation on the design and evaluation of such adaptive search engines. For example, we can use GP to learn different ranking functions for different types of queries beforehand. The search engine can intelligently select one of them and apply it to the ranking process based on the query type.

- Personalized information retrieval and delivery. An imminent problem facing search engines is the personalization of their products and the delivery of information related to the preferences of different users. Incorporating users' preferences into the product design is of primary importance in the information age. Our approach can be treated as one work along that line. The matching functions learned by our GP approach, along with the corresponding queries, can be saved in the personal profile and used for later retrieval ranking.

- Support for knowledge management. As we mentioned earlier, one of the enabling technologies for knowledge management is the search engine. Retrieval and delivery of the right and good information to the knowledge worker is one of the biggest concerns. Our approach is especially suitable for this purpose. By personalization and learning the user's preference on the relevance of documents, the same ranking function for a certain query can be used by the search engine again and again and save the knowledge worker time by eliminating the task of reviewing a large number of retrieval results. This will greatly improve the knowledge workers' productivity and help them to quickly absorb what they found.

Our future work will focus on the application of our approach to the design of an adaptive search engine, in particular, on how to design a search engine that can be tailored to the preferences of different users. We would like to refine our approach and test it in the WWW environment to see its effects.

Acknowledgements

We wish to thank the Center for the Study of Complex Systems at the University of Michigan for providing the computational resources for our experiments. We also wish to thank Professor John Holland and Professor Rick L. Riolo for many helpful discussions on the application of Genetic Algorithms/Genetic Programming to solve information retrieval problems. The programming support by Jing Xu is also greatly appreciated.

References[4]

Censorware. April 2000. http://www.censorware.org/web_size/.

Chen, H., Chung, Y., Ramsey, M., and Yang, C. "A Smart Itsy Bitsy Spider for the Web," *Journal of the American Society for Information Science* (49:7), 1998a, pp. 604-618.

Chen, H., Shankaranarayanan, G., She, L., and Lyer, A. "A Machine Learning Approach to Inductive Query by Examples: An Experiment Using Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing," *Journal of the American Society for Information Science* (49:8), 1998b, pp. 693-705.

[4]The following reference list contains URLs for World Wide Web pages. These links existed as of the date of submission but are not guaranteed to be working thereafter. The contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced. The author(s) of the Web pages, not ICIS, is (are) responsible for the accuracy of their content. The author(s) of this article, not ICIS, is (are) responsible for the accuracy of the URL and version information.

Davenport, T. H., and Laurence, P. *Working Knowledge: How Organizations Manage What They Know*, Boston: Harvard Business School Press, 1998.

Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley Publishing Co., 1989.

Gordon, M. "Probabilistic and Genetic Algorithms for Document Retrieval," *Communications of ACM* (31:2), 1988, pp. 152-169.

Gordon, M. "User-based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm," *Journal of the American Society for Information Science* (42:5), 1991, pp. 311-322.

Gordon, M., and Pathak, P. "Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines," *Information Processing and Management* (35:2), 1999, pp. 141-180.

Harman, D. "Overview of the First Text REtrieval Conference (TREC-1)," in *Proceedings of the First Text Retrieval Conference*, D. K. Harman (ed.), Washington: National Institute of Standards and Technology (NIST) Special Publication 500-207, 1993, pp.1-20.

Harman, D. "Overview of the Fourth Text REtrieval Conference (TREC-4)," in *Proceedings of the Fourth Text Retrieval Conference*, D. K. Harman(eds.), Washington: National Institute of Standards and Technology (NIST) Special Publication 500-236, 1996, pp. 1-24.

Harman, D. "Relevance Feedback Revisited," in *Proceedings of the Eleventh ACM Special Interest Group on Information Retrieval (SIGIR) Conference,* New York: ACM Press, 1992, pp. 321-331.

Holland, J. H. *Adaptation in Natural and Artificial Systems* (2nd ed.), Cambridge, MA: MIT Press, 1992.

Hull, D. "Using Statistical Testing in the Evaluation of Retrieval Experiments," in *Proceedings of the Sixteenth ACM Special Special Interest Group on Information Retrieval (SIGIR) Conference*, New York: ACM Press, 1993, pp. 329-338.

IDS. Internet Domain Survey, April 2000. http://www.isc.org/ds/.

Jones, W. P., and Furnas, G. W. "Pictures of Relevance: A Geometric Analysis of Similarity Measures," *Journal of the American Society for Information Science* (38:6), 1987, pp. 420-442.

Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* Cambridge, MA: MIT Press, 1992.

Lancaster, F. W., and Warner, A. J. *Information Retrieval Today*, Arlington, VA: Information Resources Press, 1993.

Langdon, W. B. *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming,* Boston: Kluwer Academic Publishing, 1998.

Lawrence, S., and Giles, C. L. "Accessibility of Information on the Web," Nature, (400), 1999, pp. 107-109.

Lawrence, S., and Giles, C. L. "Searching the World Wide Web," *Science* (280:3), 1997, pp. 98-100.

Martin-Bautista, M. J., Vila, M., and Larsen, H. L. "A Fuzzy Genetic Algorithm Approach to an Adaptive Information Retrieval Agent," *Journal of the American Society for Information Science* (50:9), 1999, pp. 760-771.

Mitchell, T. M. *Machine Learning*, New York: McGraw Hill, 1997.

NetSizer. April 2000. http://www.netsizer.com/.

Pathak P., Gordon, M., and Fan, W. "Effective Information Retrieval Using Genetic Algorithms Based Matching Function Adaptation," in *Proceedings of the Thirty-third Hawaii International Conference on System Science (HICSS)*, Los Alamitos, CA: IEEE Press, 2000.

Petry, F., Buckles, B., Prabhu, D., and Kraft, D. "Fuzzy Information Retrieval Using Genetic Algorithms and Relevance Feedback," in *Proceedings of the Fifty-sixth Annual Meeting of the American Society for Information Science Annual Meeting* (Vol. 30), S. Bonzi (ed.), Silver Spring, MD, 1993, pp. 122-125.

Raghavan, V. V., and Agarwal, B. "Optimal Determination of User-Oriented Clusters: An Application for the Reproductive Plan," in the *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA, 1987, pp. 241-246.

Rorvig, M. "Images of Similarity: A Visual Exploration of Optimal Similarity Metrics and Scaling Properties of TREC Topic-Document Sets," *Journal of the American Society for Information Science* (50:8), 1999, pp. 639-655.

Salton, G. *Automatic Text Processing*, Reading, MA: Addison-Wesley Publishing Co., 1989.

Salton, G., and Buckley, C. "Term Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management* (24:5), 1988, pp. 513-523.

Singhal, A., Salton, G., Mitra, M., and Buckley, C. "Document Length Normalization," *Information Processing and Management* (32:5), 1996, pp. 619-633.

Van Rijsbergen, C. J. *Information Retrieval* (2nd ed.), London: Butterworth, 1979.

Yang, J., and Korfhage, R. R. "Effects of Query Term Weights Modification in Document Retrieval: A Study Based on a Genetic Algorithm," in *Proceedings of the Second Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, NV, 1993, pp. 271-285.

Yang, J., Korfhage, R. R., and Rasmussen, E. "Query Improvement in Information Retrieval Using Genetic Algorithms: A Report on the Experiments of the TREC Project," in *Proceedings of the First Text Retrieval Conference*, D. K. Harman (ed.), Washington, DC: National Institute of Standards and Technology (NIST) Special Publication 500-207, March 1993, pp. 31-58.

Zobel, J., and Moffat, A. "Exploring the Similarity Space," *SIGIR Forum,* (32:1), 1998, pp. 18-34.