
Genetic Programming-Based Discovery of Ranking Functions for Effective Web Search

WEIGUO FAN, MICHAEL D. GORDON, AND PRAVEEN PATHAK

WEIGUO FAN is an Assistant Professor of Information Systems and Computer Science at the Virginia Polytechnic Institute and State University. He received his Ph.D. in Information Systems from the Ross School of Business, University of Michigan. His research interests focus on the design and development of novel information technologies—data mining, text/Web mining, business intelligence, personalization and knowledge management techniques—to support better business information management and decision-making. His research has been published in *Journal of Management Information Systems*, *Communications of the ACM*, *Information Processing and Management*, *IEEE Transactions on Knowledge and Data Engineering*, *Information Systems*, *Decision Support Systems*, *ACM Transactions on Internet Technology*, *Journal of Classification*, *Journal of the American Society on Information Science and Technology*, *International Journal of Electronic Business*, and in conference proceedings such as ICIS, HICSS, AMCIS, WWW, CIKM, JCDL, SIKDD, and SIGIR.

MICHAEL D. GORDON is a Professor of Business Information Technology and Associate Dean for Information Technology at the Ross School of Business, University of Michigan. He received his Ph.D. from the University of Michigan. His research interests include information retrieval, especially adaptive methods and methods that support knowledge sharing among groups; information and communication technology in the service of social enterprise (promoting economic development, providing health-care delivery, and improving educational opportunities for the poor); and using information technology along with social methods to support business education. He has published in *Journal of Management Information Systems*, *Information Systems Research*, *Communications of the ACM*, *Information Processing and Management*, *IEEE Transactions on Knowledge and Data Engineering*, *Decision Support Systems*, *ACM Transactions on Internet Technology*, and *Journal of the American Society for Information Science and Technology*.

PRAVEEN PATHAK is an Assistant Professor of Decision and Information Sciences at the Warrington College of Business at the University of Florida. He received his Ph.D. in Computer and Information Systems from the Ross School of Business, University of Michigan. His research interests include information retrieval, text mining, business intelligence, and knowledge management. His research has been published in *Journal of Management Information Systems*, *Information Processing and Management*, *IEEE Transactions on Knowledge and Data Engineering*, *Decision Support Systems*, *Journal of the American Society on Information Science and Technology*, and in conference proceedings such as ICIS, HICSS, AMCIS, and WITS.

Journal of Management Information Systems / Spring 2005, Vol. 21, No. 4, pp. 37–56.

© 2005 M.E. Sharpe, Inc.

0742-1222 / 2005 \$9.50 + 0.00.

ABSTRACT: Web search engines have become an integral part of the daily life of a knowledge worker, who depends on these search engines to retrieve relevant information from the Web or from the company's vast document databases. Current search engines are very fast in terms of their response time to a user query. But their usefulness to the user in terms of retrieval performance leaves a lot to be desired. Typically, the user has to sift through a lot of nonrelevant documents to get only a few relevant ones for the user's information needs. Ranking functions play a very important role in the search engine retrieval performance. In this paper, we describe a methodology using genetic programming to discover new ranking functions for the Web-based information-seeking task. We exploit the content as well as structural information in the Web documents in the discovery process. The discovery process is carried out for both the ad hoc task and the routing task in retrieval. For either of the retrieval tasks, the retrieval performance of these newly discovered ranking functions has been found to be superior to the performance obtained by well-known ranking strategies in the information retrieval literature.

KEY WORDS AND PHRASES: business intelligence, genetic programming, information retrieval, machine learning, ranking function, search engine, text mining, Web mining.

FINDING INFORMATION FROM THE INTERNET by utilizing Web search engines is among the top three activities on the Internet according to the Searchenginewatch.com. But it is well known [16] that finding relevant information using these search engines is a difficult task. Typically, a user has to look at a lot of documents to get only a few useful and relevant ones (low precision), or the user typically does not get all the relevant documents that he or she is looking for (low recall). The user has to refine the query several times before getting some useful results. Current search engines, such as Google and Yahoo, are quite good at finding certain specific types of information, such as person names and Web site home pages, but they are not effective when faced with a generic and comprehensive search task. Their performance, in terms of precision and recall, needs to be improved a lot.

Retrieval process involves three main subsystems: documents, queries, and the ranking functions that match information in the documents with that in the queries. Ranking functions rank the documents in a decreasing order of predicted relevance to the user. Retrieval performance is affected by factors affecting any of these three subsystems. A search engine's performance can be affected by many factors including query representation, indexing of documents and queries, controlled vocabulary, stemming, and stopping words [26]. Factors affecting document representation and query representation and manipulation have received a lot of attention in the information retrieval literature. For example, query expansion techniques, based on user feedback, have been utilized to discover user's real information needs [17, 39, 40]. Similarly, modifying document descriptions has been attempted [14, 15]. In this paper, we concentrate our efforts on the third subsystem—the ranking functions.

Traditional information retrieval systems/functions, such as Okapi BM25 (denoted at Okapi) [32] and Pivoted TFIDF [36], typically use *content-based ranking* of documents. In these methods, lexico-syntactic statistics of words in documents and queries are used for ranking purposes. Some examples of such statistics include token frequency of words in documents (*tf*), number of documents the word appears in (*df*), and document length. In the context of retrieval from the Web, two more types of ranking strategies are increasingly being used. They are *link-based ranking* and *structure-based ranking*. These strategies try to exploit the typical characteristics of Web documents.

Link-based ranking functions utilize Web interconnection information to help boost the ranking performance. The main idea in these ranking functions is that if a Web page is being linked to by many other Web pages—that is, it is highly endorsed by other Web pages—then the page that it is linked to is an authoritative page and should be ranked at the top of the retrieved list. Two of the most successful ranking functions utilizing link-based information are PageRank [3], and HITS [24]. These link-based ranking functions are especially useful to identify authoritative pages on popular topics.

Structure-based ranking functions, on the other hand, assign weights to words appearing in different structural positions, such as title, header, and anchor. They then use weighting heuristics to improve ranking performance [1].

Recent TREC (Text REtrieval Conference) evaluations [20, 21] have focused on comparing various retrieval systems that cater to the retrieval of Web documents. Various competing retrieval systems used one or more of the three main retrieval strategies mentioned above. However, it was observed in these evaluations that using link-based information does not provide much help in performance improvement as compared to using the retrieval strategy based purely on lexico-syntactic content. This is especially true about queries that are looking for new documents. It is not possible for news documents to be quickly linked by many documents. Since the link-based algorithms rely on Web pages that link to an authoritative document, newer documents get relatively low retrieval scores by such algorithms. Link-based algorithms work well for relatively stable and popular topics. In the TREC evaluations, it was also observed that some ranking strategies based purely on content alone were still among the best-performing ones. For example, Okapi was found to be very successful in these evaluations. Since link-based information was not found to be very useful in improving retrieval performance in the TREC evaluations, in this paper we will concentrate our efforts on the content-based and structure-based ranking strategies.

In order to see the effects of combining the content-based and structure-based ranking strategies, we performed a preliminary experiment using the 10GB Web data available in TREC 10. We used the three well-known ranking functions in the field of information retrieval: Okapi, Pivoted TFIDF, and Inquery. More details about these functions are available in Singhal et al. [36]. It was observed that Okapi was the best-performing ranking function when only the content information was taken into account. Moreover, when we considered information from the title of the Web documents, in addition to the information in the main body of the Web documents, we observed

that the performance increased by a small margin. This indicated that adding structural information of the Web documents may help to improve the overall search performance. It is to be noted that when we added structural information with the content information in these preliminary experiments, we simply merged text from the title with the text from the main body and then applied the term weighting strategies of the individual ranking function. Ideally, we would have liked to apply different weighting strategies (*tf*, *tf*idf*, etc.) to terms in different document structures (title, body, anchor, etc.) and eventually combine these weighted scores to rank documents. This is so because the importance of terms in different structural parts of the documents may differ significantly. For example, terms appearing in the title of a Web document are more indicative of the content of the document than terms appearing only in the body of the document. Current Okapi, Pivoted TFIDF, and Inquery formulas do not support treating different structures of the Web document differently, as they do not consider the structural information at all in their ranking. However, as noted earlier, our preliminary experiments suggested that such combination of content and structural information may yield better retrieval performance.

The above discussion led us to the main question addressed in this paper: “How can we design and discover a ranking strategy for retrieving Web documents by effectively leveraging both the content and the structural information in the Web documents?”

We answer this question by exploiting the content and structural information in the Web documents by using genetic programming (GP), an evolutionary technique specifically suited to optimize structural or functional form. The discovery framework presented in this paper is based on some prior work in the area of ranking function discovery from unstructured documents [7, 8]. However, the work presented in this paper differs significantly from the prior work in two important aspects. First, the prior work is well suited to unstructured documents. It does not take into account any structural information that may be available in documents. As noted earlier, in the Web context, significant information may be available in the different structural components of documents, such as information in the title, anchor, body, or abstract for the document. Relative importance of terms occurring in different structural parts of the documents may be different. The work presented here is well suited to structured documents. Moreover, the discovery framework presented in this paper is more flexible, as it can handle both the structured and unstructured documents. Second, in the prior work, the authors discovered ranking functions for a particular specific query (the *routing task* in information retrieval). Thus, every new query requires a new discovery process to be run. In this paper, in addition to discovering ranking functions for the routing task mentioned above, we also discover ranking functions for the *ad hoc task* in retrieval. In the ad hoc task, just a single ranking function is utilized for a group of queries rather than having a different ranking function for different queries, as in the routing task.

We believe this paper makes a threefold contribution in the area of information retrieval and specifically retrieval of Web documents. First, we will present a generic ranking function discovery methodology using GP to optimize Web search engine

performance. We will demonstrate that the retrieval performance of our methodology is significantly better than that obtained by other well-known ranking functions. Second, we will demonstrate how utilizing structural information available in Web documents affects retrieval performance in both the ad hoc and routing tasks in retrieval. Third, we will look at the effects of utilizing different fitness functions on the retrieval performance. Fitness functions, as will be discussed later, play an important part in the discovery process. We will explore whether different fitness functions will yield different retrieval performance.

Background and Related Work

IN THIS SECTION, WE WILL BRIEFLY REVIEW research related to our work in this paper. Specifically, we will first review the Vector Space Model (VSM), which is the theoretical model upon which we base our framework of ranking function discovery. Then we will review some of the work in the area of ranking function optimization in the field of information retrieval.

Vector Space Model

VSM is a theoretically well-grounded model in information retrieval. It is based on vector space and, hence, is easily interpreted from a geometric perspective [23, 34]. Each document and query is placed in an n dimensional space where its properties can be studied using geometrical similarity between the query and document vectors. This model has been one of the most successful models in various performance evaluation studies [18, 19, 33, 35], and most existing search engines and information retrieval systems are designed based on it.

In VSM, both documents and queries are represented as vectors of terms. Suppose there are t terms in the collection, then a document D and a query Q are represented as:

$$D = (w_{d1}, w_{d2}, \dots, w_{dt})$$

$$Q = (w_{q1}, w_{q2}, \dots, w_{qt}),$$

where w_{di} , w_{qi} (for $i = 1$ to t) are weights assigned to different terms in the document and query, respectively. The similarity between the two vectors is calculated as the cosine of the angle between the two vectors. It is expressed as [35]:

$$\text{Similarity}(Q, D) = \frac{\sum_{i=1}^t w_{qi} w_{di}}{\sqrt{\sum_{i=1}^t (w_{qi})^2 * \sum_{i=1}^t (w_{di})^2}}. \quad (1)$$

The score, called retrieval status value (*RSV*), is calculated for each document in the collection and the documents are ordered and presented to the user in the decreasing order of *RSV*. Various content-based features are available in VSM to compute the term weights. The most common ones are the *term frequency (tf)* and the inverse *document frequency (idf)*. Term frequency measures the number of times a term appears in a document or a query. The higher this number, the more important the term is assumed to be in describing the document. Inverse document frequency is calculated as $\log(N/DF)$, where N is the total number of documents in the collection, and DF is the number of documents in which the term appears. A high value of *idf* means the term appears in a relatively few number of documents and hence the term is assumed to be important in describing the document in which it appears. A lot of similar content-based features are available in the literature [34, 35]. The features can be combined (e.g., $tf * idf$) to generate a variety of new composite features that can be used in term weighting.

Equation (1) suggests that in order to discover a good ranking function, we need to discover the optimal way of assigning weights to document and query key words. Traditional VSM focuses on the functional space of the combination of a set of weighting features, such as *tf*, *df*, *idf*, and so on. It does not typically take into account the structural information within documents. If we qualify these weighting features to include the structural/positional information such as anchor, title, abstract, and body, we can get an expanded set of features such as tf_{anchor} , tf_{title} , $tf_{abstract}$, tf_{body} , and so on. In our discovery framework, we seek to discover new ways of leveraging structural information in assigning weights to document and query terms. The theoretical foundation serving Equation (1) can still be applied to the structural context.

Ranking Function Optimization in Information Retrieval

There has been some research in ranking function optimization in the field of information retrieval. Fuhr et al. [10, 11] used a probabilistic approach to machine learning. Their concept of *relevance description* is similar to the weighting evidence *tf*, *idf*, and so on, used in our work. However, there is a very important difference between their work and ours. In our work, we use a ranking function of arbitrary numerical functional form designed using GP, whereas in their work, the ranking function (called *retrieval function* by them) is restricted to either a polynomial regression function [10], or a logistic/log-linear function [11]. Gey [13] used similar ideas using logistic regression for ranking function design.

Some researchers have explored another line of research in the ranking function optimization by exploring a *mixture of experts* approach. In this approach, a set of ranking functions are combined using numerical methods [2, 29, 38], or by a simple majority vote [27]. In these approaches, the effectiveness of retrieval is limited by the number of ranking functions (or experts) that are used, and by the retrieval effectiveness of individual ranking functions involved. By contrast, in our approach, we can produce completely new ranking functions with significantly better performance than the individual ranking functions involved.

Chen et al. [5] proposed a hybrid genetic algorithm (GA) and neural network (NN)-based system for an information routing task. GA is used for concept extraction from a set of training documents, and NN is used for concept exploration and matching in the document collection. In this paper, we use traditional RSV relevance feedback method for query expansion, and use GP to discover optimal ranking function for information matching.

There has been some prior research in utilizing structural information for Web search [1]. Typically, this research has focused on increasing the term frequency of a term if the term appears in the title or the header of a document or if the term is in larger than the average font size. Hence, just a linear combination of the term frequencies in different structural components of the document is used. By contrast, our work focuses on not just the linear combinations of term frequencies but also looks at the nonlinear combinations of these weights tuned using GP. Our work truly discovers the best way to combine clues from different structural components of the document.

Ranking Function Discovery Methodology

IN THIS SECTION, WE DESCRIBE OUR METHODOLOGY to combine the content and structural clues available in Web documents to enhance retrieval performance. We use GP to achieve this. As mentioned in the previous section, we use VSM as the underlying theoretical model of retrieval. First, we begin with an introduction to the GP technique and the key components involved and then present in detail our ranking function discovery process for the Web documents retrieval.

Genetic Programming

GP is an inductive learning technique mimicking the principles of biological inheritance and evolution [25]. Each potential solution to the problem is represented as an *individual* in the *population* of potential solutions. Genetic transformation operators of *selection*, *reproduction*, *crossover*, and *mutation* are applied to the individuals in each generation to yield more diverse and better-performing individuals in subsequent generation. The selection and transformation process is repeated iteratively for a few generations to yield optimal or near-optimal solutions to the problem at hand. Evolutionary techniques have been tried before in the field of information retrieval [4, 15, 28, 29, 30] and in other function discovery problems [6].

Ranking Function Discovery Using Genetic Programming

In order to apply GP for ranking function discovery, we first need to have a representation of an individual in the population. We choose to represent an individual in the population using a tree structure. A sample individual in our population is as shown in Figure 1. A tree-based representation allows for ease of parsing and implementation. Moreover, this implementation is very easy to interpret.

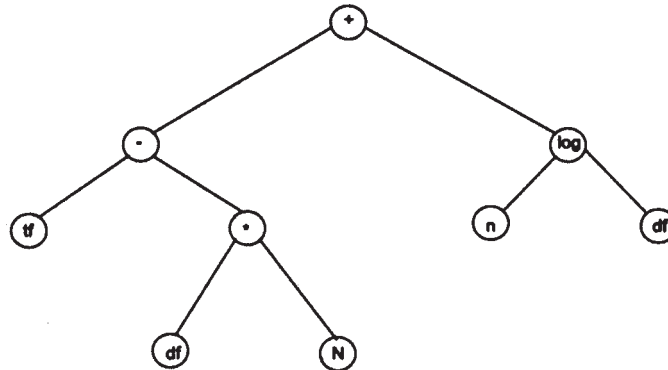


Figure 1. A Sample Tree Representation for a Ranking Function

We will use GP to discover an optimal tree for the retrieval task at hand. GP is chosen for several reasons. First, GP can be used to optimize any type of fitness/objective function. It does not require the fitness function to be continuous or differentiable. Many fitness functions in information retrieval are discrete in nature (we will talk of our fitness function later, but it also falls under this category), and hence GP is well suited for such a task. Second, GP has been shown to be very useful for nonlinear function discovery [25]. Finally, it has been empirically shown that solutions discovered by GP are typically significantly better than those discovered by other heuristic algorithms and are nearly close to the global optimum [25].

In order to apply GP in our context, we need to define several components for it. These are given in Table 1. For the purpose of our discovery framework, we will define these parameters as follows:

- An *individual* in the population is expressed in terms of a tree that represents one possible ranking function. A *population* in a *generation* consists of P such trees.
- *Terminals*: We use the features mentioned in Table 2 and real constants as the terminals. In this table, “XX” can stand for the entire document, or for the individual structural parts of the documents such as the anchor, the title, the body, or the abstract of the document. When the “XX” is used for the entire document, then we are basically capturing the *content information* in the document. Any other use of “XX” captures the *structural information* in the document. The table thus shows 42 terminals used to capture the content as well as the structural information in the Web documents.
- *Functions*: We use $+$, $-$, $*$, $/$, and \log as the functions allowed.
- *Fitness function*: We primarily use FFP4 [9] as the fitness function. It is an order-based fitness function where the fitness value depends on the order in which the relevant documents are retrieved. It was designed using utility-theoretic perspective. For the documents retrieved early in order of retrieval, this function yielded the highest utility as compared to various other fitness func-

Table 1. Essential GP Components

GP parameters	Meaning
Terminals	Leaf nodes in the tree data structure
Functions	Nonleaf nodes used to combine the leaf nodes. Typically numerical operations.
Fitness function	The objective function that needs to be optimized.
Reproduction and crossover	Genetic operators used to copy <i>fit</i> solutions from one generation to another and to introduce <i>diversity</i> in the population.

Table 2. Terminals Used in the GP Discovery Process

Features used	Statistical meaning
tf_XX	Number of times the term appeared in part XX of the document.
tf_max_XX	Maximum tf in the part XX of the document.
tf_avg_XX	Average tf in the part XX of the document.
$tf_max_XX_Col$	Maximum tf_XX in the entire document collection.
df_XX	Number of documents in the collection the term appeared in the part XX .
df_max_XX	Maximum df_XX .
N	Number of documents in the entire text collection.
$Length_XX$	Length of a document part XX .
$Length_avg_XX_Col$	Average length of part XX in the entire collection.
n	Number of unique terms in the document.

Note: XX here stands for different parts (either *entire document*, *anchor*, *title*, *body*, or *abstract*) of an HTML document.

tions tested in [9]. We use this fitness function, as it has been found to be very effective in retrieval studies.

For fitness function comparison purposes, we also use P_Avg as the fitness function. It is defined in as:

$$P_Avg = \frac{\sum_{i=1}^{|D|} r(d_i) * \left(\frac{\sum_{j=1}^i r(d_j)}{i} \right)}{TRel}, \quad (2)$$

where $r(d_i) \in (0,1)$ is the relevance score assigned to a document, being 1 if the document is relevant and 0 otherwise. $|D|$ is the total number of retrieved documents. $TRel$ is the total number of relevant documents for the query.

- *Reproduction*: Reproduction copies the top (in terms of fitness) trees in the population into the next population. If P is the population size and reproduction rate is $rate_r$, then top $(rate_r * P)$ trees are copied into next generation. $rate_r$ is set to 0.1.
- *Crossover*: We use the tournament selection to select, with replacement, six random trees from the population. The top two among the six trees (in terms of fitness) are selected for crossover and they exchange subtrees to form trees for the next generation. An illustration of a typical crossover using tree structure is as shown in Figure 2.

It is to be noted that we did not include *mutation* operator, as it has been shown [25] that it does not contribute significantly to the results when utilizing tree structure for the GP. Given the above component settings, the overall ranking function discovery framework is as shown in Figure 3. We now proceed to describe the research questions that we posed to test our ranking function discovery process for retrieving Web documents.

Research Questions/Hypotheses

VIABILITY OF OUR APPROACH IN THE WEB SEARCH CONTEXT is tested by answering the following research questions/hypotheses and conducting experiments to test them. Before we present the research questions, it is important to note that we have two different types of queries and two different types of search tasks.

The two types of queries are: user-provided queries (*short queries* in the results section) and relevance feedback queries (*feedback queries* in the results section). User-provided queries correspond to the typical Web search scenario where the search queries are very short (typically two to four terms in a user query) [22]. Feedback queries, on the other hand, are constructed automatically from the initial user-provided queries based on the relevance feedback information available in the training data set. These represent a context where users' information needs are more fully and accurately represented. Feedback queries are generated from the user-provided queries by using the Robertson Selection Value (RSV) formula [31]. This method uses relevant documents to identify the best terms for a user's query, even if the user does not include them in the query. For each query, we selected the top ten words from the relevant documents to form the feedback queries. We found this size works the best in the training collection for the three well-known ranking functions: Okapi, Pivoted Tfidf, and Inquiry [12, 36]. In both the *short* and *feedback queries*, terms were weighted by term frequency for the whole query.

In addition to the two query types, we also used two different search tasks mentioned earlier. The first task is the *ad hoc task*, where one ranking function is discovered for a set of queries. The other task is the *routing task*, where one separate ranking function is discovered for each individual query. This represents a scenario, such as information filtering, where the knowledge worker is interested in a specific topic and wants to see new documents tailored to this specific need.

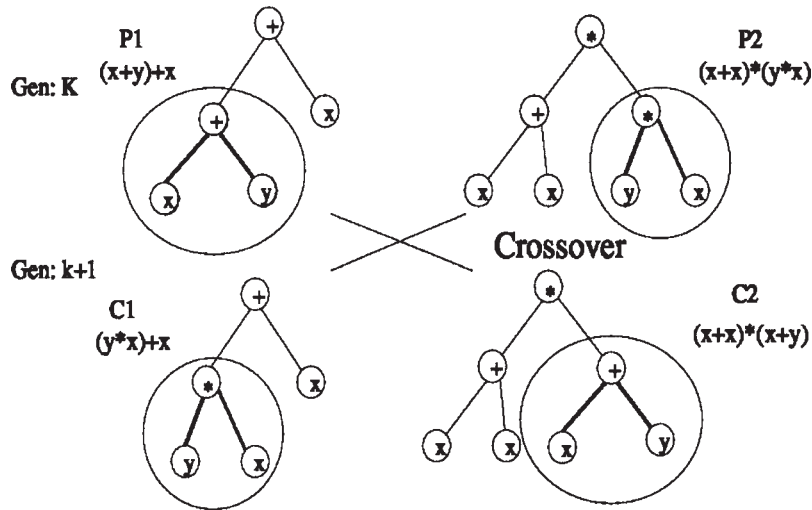


Figure 2. Crossover in Tree Representation

- Generate an initial population of random *ranking trees*
- Training phase
 - Perform following substeps on training documents for N_{gen} generations
 - Calculate the fitness of each ranking tree
 - Record the top N_{top} ranking trees with their fitness
 - Create new population by using genetic operators
 - Reproduction
 - Crossover
- Validation phase
 - Apply the recorded ($N_{gen} \times N_{top}$) candidate ranking trees on validation documents
 - Select the best performing tree as the discovery output
- Test phase
 - Apply the chosen tree in the validation phase on the test data set
 - Calculate performance measures

Figure 3. Ranking Function Discovery Process. Note: N_{gen} and N_{top} are user-specified parameters.

The following six research questions/hypotheses were tested in experiments:

1. Retrieval performance of our system (GP+S for short) incorporating GP to evolve functions utilizing both the content and the structural clues in documents is better than the performance obtained by other baseline retrieval systems for any query type (short or feedback) and any search task (ad hoc or routing).
2. Absolute retrieval performance of our system (GP+S) will increase for routing task compared to the performance for the ad hoc task for either type of queries. This is so because, in the routing task, GP adapts the function for an

individual query rather than for a group of queries as is done for ad hoc task. Hence, for an individual query, the GP should be able to better fine-tune the discovered ranking function.

3. Absolute performance will increase for all the systems (baselines and our GP+S) as we go from short query to feedback query irrespective of the search task. As stated earlier, the feedback query utilizes feedback mechanism to improve the query. Such a feedback query is better able to capture user information needs and hence should perform better than the short query that does not incorporate feedback.
4. Performance improvement by our system (GP+S) over the baselines performance is much better for short queries than for feedback queries for either type of search task. As previously stated, short queries do not incorporate feedback. But with feedback queries, because of the feedback mechanism, some of the performance gains are already achieved by any of the baselines merely by going from short queries to feedback queries. Hence, performance gains achieved by our system over the baselines are proportionately lower for feedback queries than they are for short queries.
5. Does incorporating document structure information while evolving ranking functions using GP (GP+S) help as compared to a vanilla GP implementation that does not incorporate document structure information? We believe for routing purposes our system (GP+S) should help, as this system will be better able to fine-tune ranking functions for individual queries by exploiting structural information in documents.
6. Does the choice of fitness function used in our methodology affect the retrieval performance? Based on previous work in this area [9], we believe there will be an effect of choice of fitness function. We would like to test that effect.

Experiments

TO ANSWER THE QUESTIONS RAISED in the previous section, we conducted six experiments. A combination of query type (short and feedback) and type of search task (ad hoc and routing) gives us four different experiments. We conducted two more experiments, on the ad hoc queries, to see the effect of fitness function used during the discovery process.

Data

For all the experiments, we use the Web track document collection from the TREC 9 and TREC 10 conferences [20, 21]. It has been used extensively in information retrieval evaluation studies. Residual collection method [34] is used to segment the data into three data sets: for training (50 percent), for validation (20 percent), and for testing (30 percent). There are a total of 100 queries used in these data sets. We use only 88 queries of these, since the remaining 12 do not have any relevant documents

in the validation and the test data sets. The performance results reported in the next section are based on the retrieval results obtained on test data set.

Queries Used

As stated earlier, we used 88 queries out of 100 queries available in TREC 9 and TREC 10. There has been a lot of research in information retrieval in improving query descriptions through relevance feedback. We wanted to test the effect of such feedback on the performance of our framework. Hence, we used two types of queries in the experiments. The first type is the raw queries (user-provided queries) available from TREC. The second type is the feedback query, as explained earlier.

Performance Measures

We use P_Avg performance measure to report our results, as shown in Equation (2). As mentioned earlier, it is the most widely used measure in information retrieval. It not only takes into account how many relevant documents are found but also how early in the retrieval order they are found. Thus, it is a combination of precision and recall, the two well-known retrieval measures.

Fitness Functions

We primarily use the FFP4 [9] fitness function in our experiments. This order-based fitness function has been found to be very effective in other retrieval studies. In order to see how fitness function choice affects our retrieval performance, we also conducted two minor experiments with P_Avg (Equation (2)) as the fitness function.

Genetic Programming Parameters

We ran the GP process for 20 generations (parameter N_{gen} in Figure 3). Our preliminary experiments suggested that there is no significant improvement in performance beyond 20 generations. In each generation, we recorded the top 10 ranking trees (parameter N_{top}) along with their fitness.

Baselines

We compared the retrieval results of our ranking function discovery framework with those obtained by other well-known ranking functions. Three ranking functions, Okapi, Pivoted TFIDF, and Inquiry, are used as baselines. These have been found in the literature to be most effective for Web-based retrieval [36].

In addition to these three baselines, we implemented a classifier-based ranking scheme using support vector machines (SVM) as done in Sun et al. [37]. We used

Table 3. Performance Comparisons for Ad Hoc Task for Short Query

Ranking function	Performance	
	P_{avg}	Improvement by (GP+S) (percent)
GP + Structure (GP+S)	0.3002	—
Pivoted TFIDF	0.1736	73
Inquery	0.1749	72
Okapi	0.2247	34
SVM	0.2199	37
GP (no structure)	0.2955	2

both the content and the structure information of the Web documents while training the SVM. SVM classifier was trained for each query, and the model produced was subsequently applied to the test data set to get the performance results.

Results and Discussion

IN THIS SECTION, WE PROVIDE THE RESULTS of the experiments conducted. These results are given in Tables 3 through 7 and Figures 4 and 5. Tables 3 through 6 summarize the retrieval performance in terms of P_{Avg} for six retrieval systems/functions. The last column in these tables shows the percentage improvement by our system (GP+S) over the four baselines, and over the performance obtained by vanilla GP, which does not take document structural information into account. Table 7 lists the retrieval performance when using two different fitness functions for short as well as feedback queries. Figures graphically depict the retrieval performance for the six systems. Performance for both the short and feedback queries are presented. Several interesting observations can be made about the results.

- The structural information significantly outperform the four baselines used for both the ad hoc and routing tasks in retrieval as well as for the user-provided short queries and feedback queries. The performance improvement, in terms of P_{Avg} , for the ad hoc task varies from 9 percent to 73 percent depending upon the type of query and the baseline used. Similarly, the performance improvement over the baselines for the routing task varies from 11 percent to 85 percent depending upon the type of query. All these performance improvements were statistically significant at $p < 0.05$. This validates our research question 1 in the fourth section.
- Queries as we go from routing task in retrieval (Tables 5 and 6) to ad hoc task in retrieval (Tables 3 and 4). This is expected because, by the very nature of ad hoc queries, the discovery process is trying to find ranking functions that are suitable for all the queries together, while in the case of routing queries, the discov-

Table 4. Performance Comparison for Ad Hoc Task for Feedback Query

Ranking function	Performance	
	P_{avg}	Improvement by (GP+S) (percent)
GP + Structure (GP+S)	0.4443	—
Pivoted TFIDF	0.3382	31
Inquery	0.2792	59
Okapi	0.4001	11
SVM	0.4059	9
GP (no structure)	0.4477	-1

Table 5. Performance Comparison for Routing Task for Short Query

Ranking function	Performance	
	P_{avg}	Improvement by (GP+S) (percent)
GP + Structure (GP+S)	0.3218	—
Pivoted TFIDF	0.1736	85
Inquery	0.1749	84
Okapi	0.2247	43
SVM	0.2199	46
GP (no structure)	0.2849	13

Table 6. Performance Comparison for Routing Task for Feedback Query

Ranking function	Performance	
	P_{avg}	Improvement by (GP+S) (percent)
GP + Structure (GP+S)	0.4510	—
Pivoted TFIDF	0.3382	33
Inquery	0.2792	62
Okapi	0.4001	13
SVM	0.4059	11
GP (no structure)	0.4340	4

ery process has to find ranking functions just for a particular query. Even with this decrease in performance, it is observed that the ranking functions discov-

Table 7. Effect of Fitness Function on GP+S Performance (P_Avg)

Fitness function	Query type	
	Short	Feedback
FFP4	0.3002	0.4443
P_Avg	0.2675	0.4319

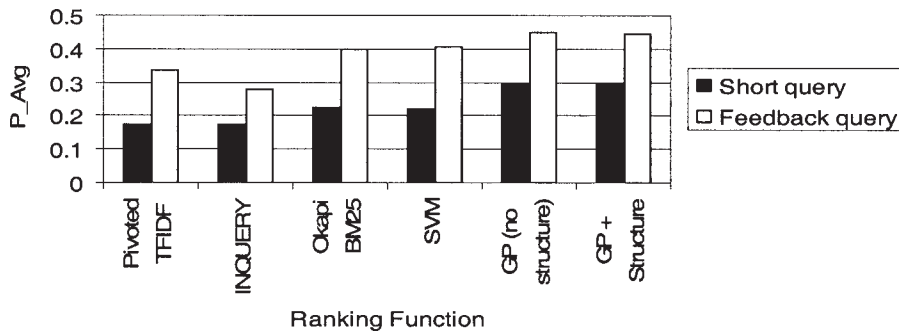


Figure 4. Retrieval Performance for Ad Hoc Task

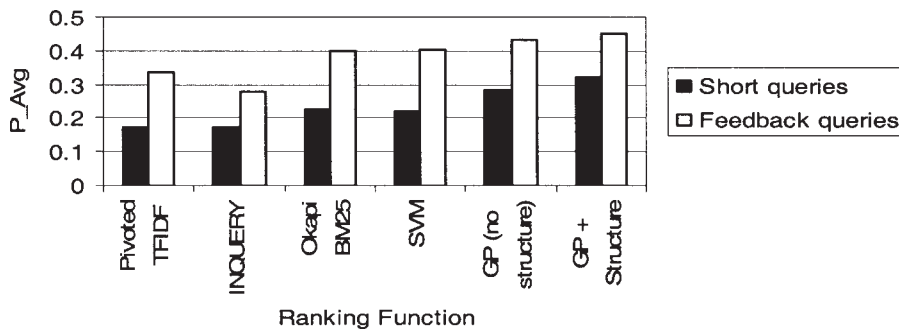


Figure 5. Retrieval Performance for Routing Task

ered by our discovery process (GP+S) still outperform the baselines (with statistical significance at $p < 0.05$). This validates our research question 2 in the fourth section.

- Baselines by our GP+S ranking function is much higher for short queries than for feedback queries. For example, for the routing task (Tables 5 and 6), performance improvement for short queries ranges from 43 percent to 85 percent, whereas the range of performance improvement for feedback queries is from 11 percent to 62 percent. Similarly, for the ad hoc task (Tables 3 and 4), the performance improvement for the short queries is from 34 percent to 73 percent, whereas it is from 9 percent to 59 percent for the feedback queries. This indicates that for

a typical Web search scenario, where the queries are short, our approach can tremendously improve performance. At the same time, even though our approach improves performance less dramatically for the feedback queries, it still significantly improves (with statistical significance) the retrieval performance over the baselines. This validates our research question 4 in the fourth section.

To answer research question 5 in the fourth section, we notice that while reporting the results in Tables 3 through 6 and Figures 4 and 5, we have introduced a ranking function called “GP (no structure).” This ranking function uses the exact same methodology for ranking function discovery as in (GP+S) ranking function except for the fact that it does not use the document structure information explicitly. Looking at Tables 3 and 4, we can see that using the document structure information explicitly in the ranking function discovery process does not give any added performance improvement if the search task is *ad hoc*. But for the *routing* search task, incorporating document structure information in the ranking function discovery process does indeed improve retrieval performance. This result has an important implication for the design of search engines: for general *ad hoc* user queries, there is no need to explicitly use document structure information. This will save the time taken to discover the ranking function using GP. But for routing task, it is still important to use the document structure information. Although it may take a little more time (than the one taken for no structure information), it will enhance retrieval performance. By the very nature of the routing task in retrieval, performance is more important to the user than slight loss in retrieval time. But for *ad hoc* retrieval task, retrieval time is important.

Any GP-based system requires a fitness function to train the discovery process. We have used FFP4 [9] as the fitness function in the results reported in Tables 3 through 6. To answer research question 6 in the fourth section, we look at the effect of using a different fitness function. We used the P_Avg (Equation (2)) as the fitness function to train the GP for the *ad hoc* retrieval task. It was observed (Table 7) that the choice of fitness function is important for the eventual retrieval performance, and especially so for the short queries. Using P_Avg as the fitness function, the retrieval performance decreased by 11 percent for short queries and by 3 percent for feedback queries. It is to be noted that although performance decreased with the use of P_Avg as the fitness function, it still is much better than the baseline performance.

A sample of the ranking function discovered by GP is

$$\log\left(\frac{tf_Doc}{tf_max_Doc} \times \frac{df_max_Doc}{df_Doc} \times \frac{length_avg_Abstract_Col}{tf_avg_Abstract}\right). \quad (3)$$

We can see that the first part of the equation, $(tf_Doc)/(tf_max_Doc)$, is the well-known normalized token frequency in information retrieval. The second part, $(df_max_DOC)/(df_Doc)$, is a new normalized inverse document frequency. The third part, $(length_avg_Abstract_Col)/(tf_avg_Abstract)$, is the structural part of the ranking function. It can be treated as a scaling factor for ranking. We can see that the GP

process is indeed able to discover well-known existing relationships as well as discover truly new ones.

Conclusion

SEARCHING FOR WEB DOCUMENTS is an ever-growing and important activity on the Internet. Unfortunately, current search engines/retrieval algorithms are inadequate in terms of retrieval performance. In this paper, we have presented a ranking function discovery framework where genetic programming is used to discover novel ranking functions for Web search. In addition to the standard document *content*-based information in the ranking process, we have also utilized the *structural* information within documents. We compared our retrieval results with those obtained by four well-known baseline retrieval functions/systems. It was found that our ranking function discovery process yielded significantly better results than the baselines for both the ad hoc and routing tasks in retrieval. We also discussed the role of fitness functions on the retrieval performance of our methodology.

We believe our work will have major implications for designing ranking functions for Web search. New ranking functions could be discovered using our methodology for different Web search contexts and for different audiences. For example, Web search queries can be grouped or clustered into different types and each type may use a different ranking function for search. Or depending on the targeted users, different individualized ranking functions could be discovered using our methodology that would lead to enhanced quality in search results. Training required in our approach can be done off-line based on either explicit or implicit user feedback (monitoring click-throughs, etc.). Such a trained ranking function now can be used for new incoming documents for routing or for retrieval for ad hoc task.

The ranking function methodology can also be used for business intelligence and security informatics where information about a particular topic is often monitored and tracked. Our methodology will enable the usage of a fine-tuned topic-specific ranking function for more effective information scanning and screening to help the user make better informed decisions.

There are many ideas to explore for future research in this promising field of inquiry. First, the newly discovered ranking functions in our approach could be used with ranking-fusion techniques to yield even better retrieval results. Second, new evidence representing additional structural information may be explored. Third, our methodology could be tried on many different document collections such as XML collections and multimedia data collections to see its viability in these settings.

REFERENCES

1. Arasu, A.; Cho, J.; Garcia-Molina, H.; Paepcke, A.; and Raghavan, S. Searching the Web. *ACM Transactions on Internet Technology*, 1, 1 (2001), 2–43.
2. Bartell, B.T.; Cottrell, G.W.; and Belew, R.K. Automatic combination of multiple ranked retrieval systems. In <<editor(s)>> *Proceedings of Seventeenth Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 1994, pp. 173–181.

3. Brin, S., and Page, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30, 1–7 (1998), 107–117.

4. Chen, H., and Kim, J. GANNET: A machine learning approach to document retrieval. *Journal of Management Information Systems*, 11, 3 (Winter 1994–95), 7–42.

5. Chen, H.; Chung, Y.; Ramsey, M.; and Yang, C. A smart itty bitsy spider for the Web. *Journal of the American Society for Information Science*, 49, 7 (1998), 604–618.

6. Dworman, G.; Kimbrough, S.; and Liang, J. On automated discovery of models using genetic programming: Bargaining in a three-agent coalitions game. *Journal of Management Information Systems*, 12, 3 (Winter 1996–97), <<page range>>.

7. Fan, W.; Gordon, M.; and Pathak, P. Discovery of context-specific ranking functions for effective information retrieval using genetic programming. *IEEE Transactions on Knowledge and Data Engineering*, 16, 4 (2004), 523–527.

8. Fan, W.; Gordon, M.; and Pathak, P. A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing and Management*, 40, 4 (2004), 587–602.

9. Fan, W.; Fox, E.A.; Pathak, P.; and Wu, H. The effects of fitness functions on genetic programming-based ranking discovery for Web search. *Journal of the American Society for Information Science and Technology*, 55, 7 (2004), 628–636.

10. Fuhr, N., and Buckley, C. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9, <<issue and/or season>> (1991), 223–248.

11. Fuhr, N., and Pfeifer, U. Probabilistic information retrieval as combination of abstraction inductive learning and probabilistic assumptions. *ACM Transactions on Information Systems*, 12, <<issue and/or season>> (1994), 92–115.

12. Gao, J.; Cao, G.; He, H.; Zhang, M.; Nie, J.; Walker, S.; and Robertson, S.E. TREC-10 Web track experiments at MSRA. In E. Voorhees and D. Harman D (eds.), *Proceedings of the Tenth Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 2002, pp. 384–392.

13. Gey, F.C. Inferring probability of relevance using the method of logistic regression. In <<editor(s)>> *Proceedings of Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM Press, 1994, pp. 222–231.

14. Gordon, M. Probabilistic and genetic algorithms for document retrieval. *Communications of the ACM*, 31, 2 (1988), 152–169.

15. 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), 311–322.

16. 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), 141–180.

17. Harman, D.K. Relevance feedback revisited. In <<editor(s)>> *Proceedings of the Eleventh ACM SIGIR Conference*. New York: ACM Press, 1992, pp. 321–331.

18. Harman, D.K. Overview of the first Text REtrieval Conference (TREC-1). In D.K. Harman (ed.), *Proceedings of the First Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 1993, pp. 1–20.

19. Harman, D.K. Overview of the fourth Text REtrieval Conference (TREC-4). In D.K. Harman (ed.), *Proceedings of the Fourth Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 1996, pp. 1–24.

20. Hawking, D. Overview of the TREC-9 web track. In E. Voorhees and D. Harman (ed.), *Proceedings of the Ninth Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 2000, pp. 86–102.

21. Hawking, D., and Craswell, N. Overview of the TREC-2001 Web track. In E. Voorhees and D.K. Harman (eds.), *Proceedings of the Tenth Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 2001, pp. 61–67.

22. Jansen, B.J.; Spink, A.; and Saracevic, T. Real life, real users, and real needs: A study and analysis of user queries on the Web. *Information Processing and Management*, 36, 2 (2000), 207–227.

23. 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), 420–442.
24. Kleinberg, J.M. Authoritative sources in a hyperlinked environment. *Journal of the Association for Computing Machinery*, 46, 5 (1999), 604 <<confirm page range>>.
25. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
26. Lancaster, F.W., and Warner, A.J. *Information Retrieval Today*. <<location>>: Information Resources Press, 1993.
27. Lee, J. Analysis of multiple evidence combination. In <<editor(s)>> *Proceedings of Twentieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM Press, 1997, pp. 267–276.
28. 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), 760–771.
29. Pathak, P.; Gordon, M.; and Fan, W. Effective information retrieval using genetic algorithms based matching function adaptation. In <<editor(s)>> *Proceedings of the Thirty-Third Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press, 2000 <<page range and/or URL where this can be found>>.
30. Raghavan, V.V., and Agarwal, B. Optimal determination of user-oriented clusters: An application for the reproductive plan. In <<editor(s)>> *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*. Cambridge, MA: <<publisher / location>>, 1987, pp. 241–246.
31. Robertson, S.E., and Jones, K.S. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27, <<issue and/or season>> (1976), 129–146.
32. Robertson, S.E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M.M.; and Gatford, M. Okapi at TREC-4. In D.K. Harman (ed.), *Proceedings of the Fourth Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 1996, pp. 73–97.
33. Salton, G. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Upper Saddle River, NJ: Prentice Hall, 1971.
34. Salton, G. *Automatic Text Processing*. Reading, MA: Addison-Wesley, 1989.
35. Salton, G., and Buckley, C. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24, 5 (1988), 513–523.
36. Singhal, A.; Salton, G.; Mitra, M.; and Buckley, C. Document length normalization. *Information Processing and Management*, 32, 5, (1996), 619–633.
37. Sun, A.; Lim, E.-P.; and Ng, W.-K. Web classification using support vector machine. In <<editor(s)>> *Proceedings of the Fourth International Workshop on Web Information and Data Management*. New York: ACM Press, 2002, pp. 96–99.
38. Vogt, C., and Cottrell, G. Fusion via a linear combination of scores. *Information Retrieval*, 1, <<issue and/or season>> (1999), 151–173.
39. Yang, J.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 D.K. Harman (ed.), *Proceedings of the First Text Retrieval Conference*. Gaithersburg, MD: National Institute of Standards and Technology, 1993, <<page range>>.
40. Yang, Y., and Pedersen, J.O. A comparative study on feature selection in text categorization. In <<editor(s)>> *Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 1997, pp. 412–420.