Ranking Function Optimization For Effective Web Search By Genetic Programming: An Empirical Study

Weiguo Fan*, Michael D. Gordon†, Praveen Pathak‡, Wensi Xi§ and Edward A. Fox¶
*Department of Accounting and Information Systems, Virginia Tech
3007 Pamplin, Blacksburg, VA 24061
Email: wfan@vt.edu. Tel: 540-231-6588
† Department of Computer Information Systems, University of Michigan
701 Tappan St, Ann Arbor, MI 48109
Email: mdgordon@umich.edu
‡Decision & Information Sciences Department, University of Florida
PO Box 117169, Gainesville FL 32611
Email: Praveen@ufl.edu
§Department of Computer Science, Virginia Tech
Blacksburg, VA 24061
Email: xwensi@vt.edu
¶Dept. of Computer Science, 660 McBryde Hall, M/C 0106, Virginia Tech
Blacksburg, VA 24061
Email: fox@vt.edu

Abstract*— Web search engines have become indispensable in our daily life to help us find the information we need. Although search engines are very fast in search response time, their effectiveness in finding useful and relevant documents at the top of the search hit list needs to be improved. In this paper, we report our experience applying Genetic Programming (GP) to the ranking function discovery problem leveraging the structural information of HTML documents. Our empirical experiments using the web track data from recent TREC conferences show that we can discover better ranking functions than existing well-known ranking strategies from IR, such as Okapi, Ptfidf. The performance is even comparable to those obtained by Support Vector Machine.

I. Introduction

Finding information on the Internet using web search engines, like Google, Yahoo, Teoma, AltaVista, is one of the top three Internet activities according to searchenginewatch.com. This fact clearly exemplifies the importance of search engines in our daily life. However, our experiences with these search engines show that their capability of getting back useful and relevant results are not always very satisfactory. We often have to refine the search query multiple times and scan through a long list of documents to find only a few of them relevant. Evaluation studies in [1] show that the current state-of-the-art search engines have not done a good job in helping user get relevant results. In fact, current search engines, such as Google, Teoma, are very effective in certain types of queries, such as name finding, homepage finding, or finding a popular topic, but not very effective for a generic and comprehensive search task where a user's query is about a specific topic. Their performance results in terms of precision and recall remain to be improved.

A search engine's performance can be affected by many factors: query representation, indexing, controlled vocabulary, stemming, stopping words, etc. [2]. But ultimately, it is affected by the ranking function, which is used to rank documents according to its match with a user's query. There are varieties of ranking functions available for web search:

- *Content-based ranking*
  These ranking functions come mainly from the traditional information retrieval fields, such as Okapi [3], Pivoted TFIDF [4]. These ranking function make extensive usage of many lexical/syntactical statistics of words in a document collection — tf, df, document length, etc. — for ranking purposes.
- *Link-based ranking*
  Link-based ranking functions utilize web interconnection information to help boost the ranking performance. Two of the most successful ranking functions are PageRank [5], HITS [6]. These link-based ranking functions are especially useful to identify those authoritative pages, which are highly endorsed by others, on popular topics.
- *Structure-based ranking*
  These types of ranking functions are commonly used by commercial search engines. They assign weights

to words appearing in different structural position, such as Title, Header, Anchor and use those weighting heuristics to improve ranking performance.

Of course, there are other ranking functions that seek to combine all the above different evidence at the content, link, and structure levels as evidenced in recent TREC web track competition [7], [8]. Several interesting observations can be made from recent TREC web track evaluations on these ranking functions:

- Using link information does not provide much help in performance improvement as compared to using content alone.
- Ranking functions based on content alone are still very successful. For example, one of the most successful ranking functions based on content only — Okapi — were found still very successful in web track evaluations [7], [8].

To further test the effects of ranking functions on search performance in a controlled setting, we did a comparative study on three content-based ranking functions — Okapi, Ptfidf, and Inquery. These three ranking functions are well-known in information retrieval field. Details of these three ranking functions can be found in [4]. We apply these ranking functions to the 10GB web data using the 50 queries from TREC 10. We summarize the performance results in Table I. As can be seen from Table I, there is a very wide variance in terms of ranking performance from the three ranking functions we compared. Okapi is clear winner among the three. Moreover, the addition of title keywords helps boost the performance for Okapi over using Body text alone by a small margin, which indicates that the structural information of HTML documents may help the overall ranking performance. One thing to be noted in Table I is that in the test of Okapi using both Title and Body texts, we simply merge these two sets of texts into one without treating these texts separately. Ideally, we want to apply different weighting schemes (tf*idf, tf, etc.) to different document structures and use one final formula to combine all the weights together. The current Okapi, Ptfidf, and Inquery formulas do not support that since they do not consider the structural information at all in their ranking.

TABLE I

PERFORMANCE COMPARISON OF THREE RANKING FUNCTIONS ON WEB DATA. THE THREE RANKING FUNCTIONS CAN BE FOUND IN [4]. THE PERFORMANCE MEASURES ARE STANDARD TREC MEASURES DESCRIBED IN MORE DETAILS IN THE EXPERIMENT SECTION OF THIS PAPER.

	P_Avg	R_P	T_Rel_Ret	P10
Okapi (Title+Body)	0.2002	0.2463	2416	0.3760
Okapi (Body)	0.1981	0.2482	2400	0.3760
Ptfidf (Body)	0.1429	0.2039	1930	0.3120
Inquery (Body)	0.1305	0.1994	1714	0.3380

This brings up our research question in this paper:

How can we design or discover a better ranking function for web-based information retrieval by effectively leveraging both content and structure information of web documents?

To answer this question, we design a novel ranking function discovery framework for web context based on our prior work on ranking function discovery on unstructured data [9], [10], [11]. This new framework differs from our previous work in that our earlier work on ranking function discovery does not make use of any structural information such as Title, Anchor, Body, Abstract, as does in the current framework. In fact, the new framework is a superset of the previous one and can be used for both structured/semi-structured and unstructured documents, while the previous framework can only be applied to unstructured ones. Moreover, we follow a new experimental design strategy in this paper. That is, we are going to optimize the ranking function for a group of queries instead of individual queries as in our previous work. In this regard, this work can be classified as the consensus ranking function discovery for web search [12].

Our paper is organized as follows. In section 2, we describe the required background information on our theoretical foundation — Vector Space Model — for information retrieval. In Section 3, we present our new ranking function discovery framework for web search context using both the content and structural information. We conduct two experiments to evaluate this framework and summarize the experimental findings in Section 4. Section 5 discusses the related works to this study and Section 6 concludes this paper and point out future research directions.

II. BACKGROUND AND THEORETICAL FOUNDATION

As we mentioned earlier, the objective of a ranking function is to match documents or information to a user's query and place them in descending order of their predicted relevance to a user's information requirement.

To facilitate this relevance estimation process, both these documents and a user's information need to be transformed into a form that can be effectively processed by computers. One of the most successful models is the so-called Vector Space Model (VSM) [13], [14].

The VSM is chosen to be the theoretical foundation for this study for two reasons:

1) *Ease of interpretation*
   The VSM is a theoretically well-grounded model. It is based on Vector Space and thus can be easily interpreted from a geometric perspective [15], [14]. For example, each document and query vector can be placed in an Euclidean n dimensional space. The properties of these two vectors, such as their similarity and closeness, can then be studied.
2) *Great success in performance evaluations*
   The VSM has been one of the most successful models in various performance evaluation studies

[16], [17], [13], [18]. Most existing search engines and information retrieval systems are designed based on this model.

More specifically, both documents and user queries are represented as vectors in the VSM. Suppose there are total *t* index terms in an entire collection, a given document *D* and query *Q* can be represented as follows:

$$D = (w_{d1}, w_{d2}, w_{d3}, \ldots, w_{dt})$$
$$Q = (w_{q1}, w_{q2}, w_{q3}, \cdots, w_{qt})$$

where w_{di}, w_{qi} (*i*=1 to *t*) are term weights assigned to different terms for the document D and query Q respectively. The similarity between a query and a document can be calculated by the widely used Cosine measure [18]:

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

Documents are then ordered by decreasing values of this measure.

There are various features (weighting evidence[1]) available in the Vector Space model to compute the term weights — w_{di}, w_{qi} (*i*=1 to *t*). One of the most widely used features for term weighting is *Term Frequency* (TF), which measures the number of times a term appears in a document or query. Another commonly used feature, measuring the rarity of a term in a collection, is the *Inverse Document Frequency* (IDF) which can be calculated by $\log(N/DF)$, where *N* is the total number of documents in a text collection, and *DF* is another feature that measures the number of documents in which a term has appeared in an entire document collection. More features used in term weighting can be found in [18], [14]. These features can also be combined to generate a wide range of new composite weighting features, e.g., TF*IDF, etc.

Note that the normalization factor in the denominator of Equation (1) is often omitted in calculation for performance reasons. In this case, the Cosine measure is replaced by the inner dot product, which can be represented as follows:

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

Because of the duality of the model between w_{qi} and w_{di} shown in Equation 2, if we merge w_{qi} into w_{di} and set $w_{qi} = 1$, Equation 2 can be further reduced to

$$Similarity(Q, D) = \sum_{i \in Q} w_{di} \qquad (3)$$

where Q is the set of keywords used in a user query.

[1]We use features and evidence interchangeably in this paper.

Equation 3 basically tells us that in order to discover a good ranking function, we need to discover the optimal way of assigning weights to document keywords. In traditional VSM, though, w_{di} is not designed to take consideration of the structural information. Instead, it focuses on the functional space of the combination of a set of weighting features, such as tf, df, idf, etc. as we mentioned above. If we qualify these weighting features to include the structural/positional information such as Anchor, Title, Abstract (the top 50 words in <Body> part of a HTML document), and Body, we get an expanded set of features including tf_{anchor}, tf_{title}, $tf_{abstract}$, tf_{body}. The theoretical foundation serving Equation 3 can still be applied to the structural context.

We use Equation 3 as the theoretical foundation for this paper. We will seek to discover new ways of leveraging structural information in assigning weights to documents terms to improve the overall ranking performance.

III. A RANKING FUNCTION DISCOVERY METHODOLOGY BASED ON GENETIC PROGRAMMING (GP)

In this Section, we propose genetic programming (GP) to study the problem of ranking function discovery in a web search context. This approach will help us automate the ranking function design by effectively leveraging both content and structure information embedded in HTML documents. We do not consider link information in this study[2]. We focus our discussion on Vector Space Model only since it is one of the most popular and successful models in IR as mentioned in the theoretical background Section. We begin with a brief introduction to the learning technique — Genetic Programming and its key components, then we present the detailed ranking function discovery framework.

Genetic Programming (GP), an extension of Genetic Algorithms (GAs), is an inductive learning[3] technique designed following the principles of biological inheritance and evolution [19]. In GP, each potential solution is called an individual in a population. An individual in GP systems is typically represented using a tree structure as shown in Figure 1.

GP works by iteratively applying genetic transformations, such as reproduction, crossover, mutation, to a population of individuals to create more diverse and better performing individuals in subsequent generations. Both GP and GA have been applied to information retrieval field [20], [9], [21], [22], [23], [24], [25].

In order to apply GP to the problem of ranking function discovery, several required key components of a GP system need to be defined. Table II lists these essential components along with their descriptions.

[2]It is not very difficult to combine our discovered ranking function with link-based evidence using probabilistic framework. We leave this for future research.

[3]learning and generalization from specific examples

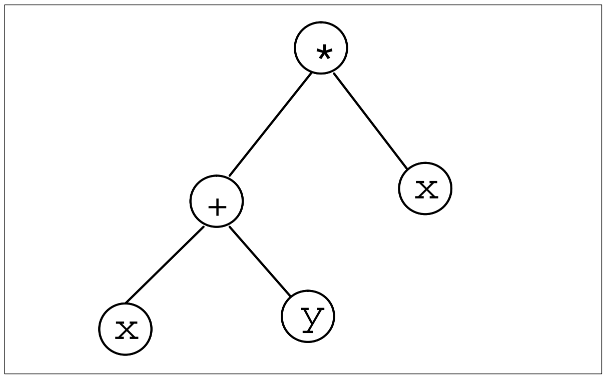

Fig. 1. A sample tree representation for a ranking function

TABLE II
ESSENTIAL GP COMPONENTS.

Components	*Meaning*
Terminals	Leaf nodes in the tree structure. i.e. x, y as in Figure 1.
Functions	Non-leaf nodes used to combine the leaf nodes. Commonly numerical. operations: +, -, *, /
Fitness function	The objective function GP aims to optimize.
Reproduction	A genetic operator that copies the individuals with the best fitness. values directly into the population of the next generation without going through the crossover operation.
Crossover	A genetic operator that exchanges sub-trees from two parents to form two new children. Its aim is to improve the diversity as well as the genetic fitness of the population. This process is shown in Figure 2.

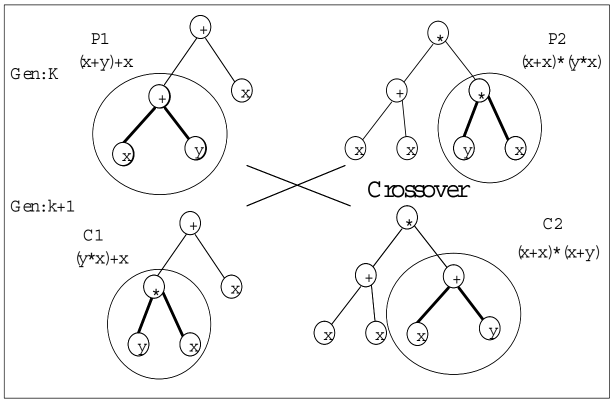

Fig. 2. A graphical illustration of the crossover operation

We set up the configurations of the GP system used for ranking function discovery as shown in Table III.

TABLE III
MODELING SETUP FOR RANKING FUNCTION DISCOVERY BY GP. REFER TO TABLE II FOR EXPLANATIONS OF THE VARIOUS COMPONENTS.

Terminals	We use features shown in Table IV as terminals.
Functions	$+$, $\times$, $/$, log
Fitness function	The average of p_avg for m queries, where p_avg is defined as p_avg = $\sum_{i=1}^{TRel} P_i/TRel$, $P_i = i/Rank_i$ for each query, $TRel$ is the total number of relevant documents for a given query, $Rank_i$ is the ranking position for the ith relevant document.
Genetic operator	Reproduction, Crossover

TABLE IV
TERMINALS USED IN OUR GP SYSTEM. X IS USED TO STAND FOR DIFFERENT PARTS OF A HTML DOCUMENT: ENTIRE DOCUMENT (DOC), ANCHOR, TITLE, BODY, ABSTRACT.

Terminals	Statistical Meaning
tf_X	The number of times a term appeared in the part X of a document
tf_max_X	The maximum tf in the part X of a document
tf_avg_X	The average tf in the part X of a document
tf_max_X_Col	The maximum tf_X in the entire document collection
df_X	The number of unique Xs a term appeared in a collection
df_max_X	The maximum df_X for a given query
N	The total number of documents in the entire text collection
length_X	The length of a document part X
length_avg_X_Col	The average length of part X in the entire collection
R	A real constant number randomly generated by the GP system
n	The number of unique terms in a document

With the above settings, the overall ranking function discovery framework is shown in Figure 3. Note that the framework described in Figure 3 differs from our previous one [9], [10], [11] in that

a) Compared to [11], we use a much larger set of terminals (features) so we can leverage structural information.
b) The fitness evaluation of each ranking tree is done at the level of multiple queries. In other words, we apply a ranking tree to multiple queries simultaneously and calculate the fitness value using the aggregated performance for all queries. In our previous work [9], [10], this was done at the individual query level. Thus the work reported in [9], [10] is more suitable for query-specific ranking or personal ranking [12]. The work reported in this paper is more suitable for ad-hoc information retrieval, or consensus ranking [12].

```
1.    Generate an initial population of random
      "ranking trees";
2.    Perform the following sub-steps on training
      documents for $N_{gen}$ generations
      2.1   Use each ranking tree to score and rank
            the collection separately for multiple
            queries
      2.2   Calculate the fitness of each ranking tree
      2.3   Record the top $N_{top}$ ranking trees
      2.4   Create new population by:
            a) Reproduction
            b) Crossover
3.    Apply the recorded ($N_{gen} \times N_{top}$) candidate
      "ranking trees" on a set of validation documents
      and select the best performing tree as the unique
      discovery output
```

Fig. 3. Overall Ranking Function Discovery Framework. N_{gen} and N_{top} are user-specified parameters.

IV. Experiments

We conduct two experiments to test the viability of the new ranking function discovery framework in the web search context. The first experiment is to use user-provided queries to see whether we can discover a better ranking function for ad-hoc information retrieval in general. This corresponds to the web search scenario where the search queries are very short (typically 2-4 words in a user query) [26]. The second experiment is to use relevance feedback queries (constructed automatically from the training collection with relevance judgment) to see whether the same framework can be applied in a context where users' information needs are more fully and accurately represented. The combination of these two experiments should provide us with insights about the capability of the GP-based discovery framework.

A. Data

For both experiments, we use the standard 10GB Web Track collection from recent TREC 9 and TREC 10 conferences [7], [8]. The same collection has been used extensively to evaluate various web-based information retrieval systems. Because our framework utilizes machine learning techniques, we use the residual collection method [14] to divide the entire data into three parts: training (50%), validation (20%) and test data (30%). The training data, along with the relevance information for queries are used by the GP-based ranking function discovery framework to generate a set of "candidate" consensus ranking functions for multiple queries (Steps 1 and 2, Figure 3). The validation data is used to choose the one that is of the best generalization capability for new data (Step 3, Figure 3). The performance comparison of all systems are based on the results on the test data only.

There are 100 topics provided in TREC 9 and 10 web track. Since 12 of these queries do not have any relevant documents in either the validation and test data set, we exclude them and use the rest 88 queries as the test queries for our experiments. All of these 88 queries have their relevance information available.

B. Performance measures

Table V lists the performance measures used in this study. All of these measures are standard performance measures used in TREC for cross-system performance comparison. These measures are selected to balance both precision[4] and recall[5]. Among the four measures, Pavg and P10 are the primary two we focus on. Pavg is a hybrid measure of average precision and recall [16], [17]. P10 is the measure used primarily in the web search context to show how good a ranking system at returning relevant documents at the top of a hit list [7], [8].

TABLE V
Performance measures and their definitions

Measure	Definition
Pavg	The average of precisions every time a new relevant is found, normalized by the total number of relevant documents in an entire collection. An equivalent mathematical definition can be found in Table III.
P10	The precision in the top 10 retrieved documents
RP	The precision when T_Rel documents are retrieved.
TRR	The total number of relevant documents retrieved for a given query

C. Baselines

In order to demonstrate the efficacy of the ranking functions discovered by GP, we need to compare them with other well-known ranking functions/systems. Three content-based ranking functions — Okapi BM25 (denoted as Okapi), Pivoted TFIDF (denoted as Ptfidf), and Inquery — are used as the baseline systems. The details of these three ranking functions can be found in [4]. In addition, we implement a classifier-based ranking scheme using Support Vector Machine (SVM) as in [27]. This ranking scheme uses both content and structure information of the HTML documents and serves as a very competitive baseline to compare against the ranking functions discovered using our ranking function discovery framework. In order to obtain meaningful results for comparison, we train a SVM classifier for each query, and the model produced is subsequently applied to the test data to obtain the performance result. We repeat this step for all 88 queries. These four systems will be used as the baselines in this study.

[4]The proportion of the retrieved documents that are relevant.
[5]The proportion of relevant documents that are retrieved.

D. Experiment 1: User-provided queries

The performance comparison results between the best ranking functions we discovered using GP-based framework (denoted GP+S) and the four baselines are summarized in Table VI. Note that Column 6 provides a comparison based on using the GP approach without structural information.

TABLE VI

PERFORMANCE COMPARISON BETWEEN SYSTEMS FOR USER-PROVIDED QUERIES. COLUMNS 2-7 ARE DIFFERENT RANKING SYSTEMS. EACH ROW REPORTS THE RESULTS FOR THE PERFORMANCE MEASURE SHOWN IN COLUMN 1. SEE TABLE V FOR THE DEFINITION OF THESE MEASURES

	Okapi	Ptfidf	Inquery	SVM+S	GP	GP+S
Pavg	0.2247	0.1736	0.1749	0.2199	0.2193	0.2675
P10	0.2420	0.2080	0.2227	0.2398	0.2409	0.2841
RP	0.2432	0.1918	0.2004	0.2353	0.2343	0.2713
TRR	16.4	15.5	14.5	16.7	16.5	16.9

As can be seen from Table VI, GP+S, clearly outperforms all the baseline systems in all four measures. Among the four baselines, SVM+S and Okapi perform the best. The difference between SVM+S and Okapi are negligible. In terms of performance improvement, GP+S outperforms Okapi by almost 20% in Pavg, 17.4% in P10. We consider such improvement quite significant.

Another interesting observation related to focus of this study is the advantage of leveraging the structural information in ranking function discovery. The performance of the best ranking function discovered using only content without consideration of structure (denoted as GP in Table VI) is clearly inferior to that of the content+structure approach. This indicates that utilizing structural information of query words in ranking function discovery is beneficial.

The best ranking function corresponding to "GP+S" in Table VI is shown as follows:

$$log\left(\frac{tf\_Doc}{tf\_max\_Doc} \times \frac{df\_max\_Doc}{df\_Doc} \times \frac{length\_avg\_Abstract\_Col}{tf\_avg\_Abstract} \right) \quad (4)$$

As we can see from Formula (4), $\frac{tf\_Doc}{tf\_max\_Doc}$ is the well-known normalized tf in IR [14]. $\frac{df\_max\_Doc}{df\_Doc}$ is a new normalized IDF. $\frac{length\_avg\_Abstract\_Col}{tf\_avg\_Abstract}$ is the structural part of the ranking function, and can be treated as the scaling factor for the ranking. In other words, larger tf_avg in the Abstract part (top 50 words within <Body> part) will lead to lower ranking score.

To see how the structural part affects the ranking performance, we drop this part and use the rest of the formula for ranking. The performance in Pavg is down to 0.11 (from 0.2675 using the full formula). This indicates that the structural component plays a substantial role in Formula 4 for the HTML ranking.

E. Experiment 2: Relevance feedback queries

For this experiment, we replace the user-provided queries with feedback queries which are generated using the RSV formula [28]. This method uses relevant documents to identify the best terms for a user's search, even if the user doesn't include them in the query. We select the top 10 words for each query as we find this size works the best in the training collection for three content-based ranking systems: Okapi, Ptfidf and Inquery. The experimental results on these feedback queries are summarized in Table VII.

TABLE VII

PERFORMANCE COMPARISON BETWEEN SYSTEMS FOR RELEVANCE FEEDBACK QUERIES. COLUMNS 2-7 ARE DIFFERENT RANKING SYSTEMS. EACH ROW REPORTS THE RESULTS FOR THE PERFORMANCE MEASURE SHOWN IN COLUMN 1. SEE TABLE V FOR THE DEFINITION OF THESE MEASURES

	Okapi	Ptfidf	Inquery	SVM+S	GP	GP+S
Pavg	0.4001	0.3382	0.2792	0.4059	0.4275	0.4319
P10	0.425	0.3614	0.3261	0.4102	0.4318	0.4341
RP	0.3928	0.3448	0.301	0.4153	0.4089	0.4063
TRR	17.3	16.6	15.4	17.1	17.3	17.3

If we compare Table VII with Table VI, one noticeable difference is the huge improvement of the performance in absolute values. The feedback queries improve the performance of the baseline systems by 80-100%. This indicates the inadequency of the information need representation in the web search context.

With the highly improved baseline results using the feedback queries, our ranking discovery framework still works very well. The newly discovered ranking function using both content and structure (GP+S) again outperforms all the baseline systems and its content-only counterpart. The improvement of GP+S over Okapi is almost 9%.

Another observation can be made with regards to Table VII is that the advantage of using structural information is less obvious than in Table VI. This can be seen from the shrinking gap between GP+S with GP. One possible explanation is that the performance gain of these feedback queries has offset the gain in using the structural information.

The best ranking function discovered by the ranking discovery framework for feedback queries is shown in Figure 4. As can be seen, this ranking function is of much more complexity than the one shown in Formula (4) for short queries. Some of the well-known term weighting strategies such as idf – N/df_Doc, normalized tf – tf_Doc/tf_avg_Body, are found useful in the ranking. There are other weighting strategies that are new to us in Figure (4)— this may be the advantage of knowledge discovery through GP.

```
   Log((((tf_avg_Anchor_Col + tf_avg_Abstract_Col)* (Log(((Log((N / tf_Doc))
*  df_max_Body_Col) / (tf_max_Doc / tf_max_Abstract)))
+  Log(df_max_Title_Col))) / ((tf_avg_Body / N)
*  (df_Doc * (n / tf_Doc)))) + Log((Log((length_avg_Abstract_Col / tf_max_Abstract))
*  ((df_max_Anchor_Col * tf_Doc) / tf_Body)))))
```

Fig. 4. The best ranking function discovered by GP for feedback queries

V. Related work

There have been several efforts on ranking function optimization in IR literature.

The earliest work is done by N. Fuhr et al. [29], [30] using probabilistic models as machine learning approaches. The concept of *relevance description* used in [29], [30] is very similar to the weighting evidence (tf, df, $\cdots$) we used for ranking. An important difference between our work from theirs is that we use a ranking function of arbitrary numerical functional form designed from GP, while in [29], [30], the ranking function (called *retrieval function* in [29], [30]) is either a polynomial regression function [29], or logistic regression/loglinear function [30]. Similar ideas using logistic regression for ranking function design and optimization have also been explored in [31].

Another line of research on ranking function optimization is following the mixture of experts approach, in which a set of ranking functions are combined either numerically through linear combination [32], [33], [25], or simple majority vote [34]. The effectiveness is limited by the number of experts (ranking functions) they used and how effective they are individually. Our work, in fact, can produce new ranking functions with better performances than existing ones. These newly discovered ranking functions can be combined using the mixture of experts approach with other well-known ranking functions to further improve the ranking performance.

VI. Discussions and future research

In this paper, we consider the problem of consensus ranking function discovery in the context of web search. A framework for tackling this problem based on Genetic Programming has been proposed and tested. Our experimental results on two different sets of queries using a very large web collection have demonstrated that the framework can be used to discover better ranking functions than existing ones. This framework can be used for optimizing ranking functions used in search engines or digital libraries. We also find that the structural information of query words are especially beneficial for ranking function discovery using short queries.

The computation time for each learning experiment (involving 10 different independent runs using different random seeds to improve the chances of finding optimal solutions) took about 5 days to finish on a Linux machine with a 1.3GHz CPU. The learning process can be further speeded up by using multiple processors, parallel processing or efficient data sampling. We consider this computation time is acceptable with the performance results we obtained.

Our future work is to test this framework on more document collections to see its viability. We also plan to test this framework at the individual query level to see the structural effects on ranking function discovery. It may be interesting to see how the approach using logistics regression as in [31] can fare against GP in these two learning tasks. Finally, new terminals (features) representing additional structural information may be explored.

References

[1] M. Gordon and P. Pathak, "Finding information on the world wide web: the retrieval effectiveness of search engines," *Information Processing and Management*, vol. 35, no. 2, pp. 141–180, 1999.

[2] F. W. Lancaster and A. J. Warner, *Information Retrieval Today*. Information Resources Press, 1993.

[3] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-4," in *Proceedings of the Fourth Text Retrieval Conference*, D. K. Harman, Ed. NIST Special Publication 500-236, 1996, pp. 73–97.

[4] A. Singhal, G. Salton, M. Mitra, and C. Buckley, "Document length normalization," *Information Processing and Management*, vol. 32, no. 5, pp. 619–633, 1996.

[5] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998. [Online]. Available: citeseer.nj.nec.com/brin98anatomy.html

[6] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999. [Online]. Available: citeseer.nj.nec.com/kleinberg99authoritative.html

[7] D. Hawking, "Overview of the TREC-9 web track," in *Proceedings of the Ninth Text Retrieval Conference*, E. M. Voorhees and D. K. Harman, Eds. NIST Special Publication 500-249, 2001, pp. 86–102.

[8] D. Hawking and N. Craswell, "Overview of the TREC-2001 web track," in *Proceedings of the Tenth Text Retrieval Conference*, E. M. Voorhees and D. K. Harman, Eds. NIST Special Publication 500-250, 2001, pp. 61–67.

[9] W. Fan, M. D. Gordon, and P. Pathak, "Personalization of search engine services for effective retrieval and knowledge management," in *Proceedings of 2000 International Conference on Information Systems (ICIS)*, Brisbane, Australia, 2000, pp. 20–34.

[10] ——, "Discovery of context-specific ranking functions for effective information retrieval using genetic programming," *IEEE Transactions on Knowledge and Data Engineering*, 2003, in press.

[11] ——, "A generic ranking function discovery framework by genetic programming for information retrieval," *Information Processing and Management*, 2003, in press.

[12] J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, "Personalized search," *Communications of the ACM*, vol. 45, no. 9, pp. 50–55, September 2002.

[13] G. Salton, *The SMART Retrieval System: Experiments in Automatic Document Processing*. New Jersey: Prentice Hall, 1971.

[14] ——, *Automatic Text Processing*. Reading, MA: Addison-Wesley Publishing Co., 1989.

[15] W. P. Jones and G. W. Furnas, "Pictures of relevance: a geometric analysis of similarity measures," *Journal of the American Society for Information Science*, vol. 38, no. 6, pp. 420–442, 1987.

[16] D. K. Harman, “Overview of the first text retrieval conference (TREC-1),” in *Proceedings of the First Text Retrieval Conference*, D. K. Harman, Ed. NIST Special Publication 500-207, 1993, pp. 1–20.

[17] ——, “Overview of the fourth text retrieval conference (TREC-4),” in *Proceedings of the Fourth Text Retrieval Conference*, D. K. Harman, Ed. NIST Special Publication 500-236, 1996, pp. 1–24.

[18] G. Salton and C. Buckley, “Term weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.

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

[20] H. Chen, Y. Chung, M. Ramsey, and C. Yang, “A smart itsy bitsy spider for the web,” *Journal of the American Society for Information Science*, vol. 49, no. 7, pp. 604–618, 1998.

[21] M. Gordon, “Probabilistic and genetic algorithms for document retrieval,” *Communications of ACM*, vol. 31, no. 2, pp. 152–169, 1988.

[22] ——, “User-based document clustering by redescribing subject descriptions with a genetic algorithm,” *Journal of the American Society for Information Science*, vol. 42, no. 5, pp. 311–322, 1991.

[23] V. V. Raghavan and B. Agarwal, “Optimal determination of user-oriented clusters: An application for the reproductive plan,” in *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA, 1987, pp. 241–246.

[24] M. J. Martin-Bautista, M. Vila, and H. L. Larsen, “A fuzzy genetic algorithm approach to an adaptive information retrieval agent,” *Journal of the American Society for Information Science*, vol. 50, no. 9, pp. 760–771, 1999.

[25] P. Pathak, M. Gordon, and W. Fan, “Effective information retrieval using genetic algorithms based matching function adaptation,” in *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS)*, Hawaii, USA, 2000.

[26] B. J. Jansen, A. Spink, and T. Saracevic, “Real life, real users, and real needs: a study and analysis of user queries on the web,” *Information Processing and Management*, vol. 36, no. 2, pp. 207–227, 2000.

[27] A. Sun, E.-P. Lim, and W.-K. Ng, “Web classification using support vector machine,” in *Proceedings of the fourth international workshop on Web information and data management*. ACM Press, 2002, pp. 96–99.

[28] S. Robertson and K. S. Jones, “Relevance weighting of search terms,” *Journal of the American Society for Information Science*, vol. 27, pp. 129–146, 1976, reprinted in: P. Willett (ed.), Document Retrieval Systems. Taylor Graham, 1988. (pp 143-160).

[29] N. Fuhr and C. Buckley, “A probabilistic learning approach for document indexing,” *ACM Transactions on Information Systems*, vol. 9, no. 3, pp. 223–248, 1991. [Online]. Available: citeseer.nj.nec.com/fuhr91probabilistic.html

[30] N. Fuhr and U. Pfeifer, “Probabilistic information retrieval as combination of abstraction, inductive learning and probabilistic assumptions,” *ACM Transactions on Information Systems*, vol. 12, no. 1, pp. 92–115, 1994. [Online]. Available: citeseer.nj.nec.com/fuhr94probabilistic.html

[31] F. C. Gey, “Inferring probability of relevance using the method of logistic regression,” in *the Proceedings of Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994, pp. 222–231.

[32] B. T. Bartell, G. W. Cottrell, and R. K. Belew, “Automatic combination of multiple ranked retrieval systems,” in *the Proceedings of Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994, pp. 173–181. [Online]. Available: citeseer.nj.nec.com/bartell94automatic.html

[33] C. C. Vogt and G. W. Cottrell, “Fusion via a linear combination of scores,” *Information Retrieval*, vol. 1, no. 3, pp. 151–173, 1999.

[34] J. H. Lee, “Analyses of multiple evidence combination,” in *the Proceedings of Twentieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1997, pp. 267–276.