A Combined Component Approach for Finding Collection-Adapted Ranking Functions based on Genetic Programming

Humberto Mossri de Almeida[1] Marcos André Gonçalves[1]
Marco Cristo[2] Pável Calado[3]

[1]Federal Univ. of Minas Gerais
Dept. of Computer Science
Belo Horizonte, Brazil
{hmossri, mgoncalv}@dcc.ufmg.br

[2]FUCAPI - Analysis, Research and Tech. Innovation Center
Manaus, Brazil
marco.cristo@fucapi.br

[3]IST/INESC-ID
Lisboa, Portugal
pavel.calado@tagus.ist.utl.pt

ABSTRACT

In this paper, we propose a new method to discover collection-adapted ranking functions based on Genetic Programming (GP). Our *Combined Component Approach* (CCA) is based on the combination of several term-weighting components (i.e., term frequency, collection frequency, normalization) extracted from well-known ranking functions. In contrast to related work, the GP terminals in our CCA are not based on simple statistical information of a document collection, but on meaningful, effective, and proven components. Experimental results show that our approach was able to outperform standard TF-IDF, BM25 and another GP-based approach in two different collections.

CCA obtained improvements in mean average precision up to 40.87% for the TREC-8 collection, and 24.85% for the WBR99 collection (a large Brazilian Web collection), over the baseline functions. The CCA evolution process also was able to reduce the overtraining, commonly found in machine learning methods, especially genetic programming, and to converge faster than the other GP-based approach used for comparison.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms, Measurement, Experimentation.

Keywords: Information Retrieval, Ranking Functions, Term-weighting, Genetic Programming, Machine Learning.

1. INTRODUCTION

The growth in volume of the Web and other textual repositories, such as digital libraries, throughout the last decade, has made the information retrieval task difficult, costly, and in many cases, very complex for the end user. In this context, search engines became valuable tools to help users find content relevant to their information needs. Naturally, research on information retrieval models that can effectively rank search results according to document relevance has become a fundamental subject.

Information Retrieval models have come a long way. Although the most popular is still undoubtedly the vector space model proposed by Salton [19], many new or complementary alternatives have been proposed, such as the Probabilistic Model [16]. From all these models, document ranking formulas can be derived for document searching.

Thus, many alternatives exist on how to compose a ranking function. Most of them have a common characteristic: they attempt to be very general in nature, i.e., they were designed to be applied in any type of collection. The work of Zobel and Moffat [26], for example, presented more than one million possibilities to compute a similarity function. However, after all the experiments, they concluded that no weighting scheme is consistently good in all collections. That is, a ranking function can have success in one domain but fail in another. Further, they comment that it would be prohibitive to discover the best weighting scheme simply by an exhaustive exploration of the similarity space.

In this work, we discover specialized ranking strategies for specific collections. Our method is able to consider the important and unique characteristics of each collection so that the discovered function is more effective than any general solution. To accomplish this, we use Genetic Programming (GP), a machine learning technique inspired by Darwinian evolutionary processes, to discover specific ranking functions for each document collection. GP has been successful in many IR problems [7–10, 13, 22]. GP was chosen due to its ability to find any arbitrary function, even when dealing with very large search spaces. However, differently from other GP-based approaches, which use only basic statistical information from terms and documents, our strategy uses rich, meaningful, and proven effective components present in well-known ranking formulas, such as Okapi BM25 [18] and Pivoted TF-IDF [21]. Our assumption is that by providing these human-discovered formula components as building blocks, the GP process can take advantage of all the human

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.

knowledge that has been applied to produce them. As a consequence, it will be able to better explore the search space.

To validate our GP approach we performed experiments with the TREC-8 and WBR99 collections. Results indicate that the use of meaningful components in a GP-based framework leads to effective ranking functions that significantly outperform the baselines (standard TF-IDF, BM25 and another GP-based approach [9]). Our Combined Component Approach (CCA) ranking functions also converged to good results faster than the GP approach used as baseline, and the overtraining also was reduced.

This paper is organized as follows. In Section 2, we provide background information on term-weighting components and genetic programming. In Section 3, we present our Combined Component Approach for similarity calculation. The collections used and experimental results are detailed in Section 4. In Section 5, we describe related work. Finally, Section 6 concludes the paper and gives suggestions for future work.

2. BACKGROUND

In this section we present the term-weighting components used in our approach and a brief review of some concepts of Genetic Programming.

2.1 Term-Weighting Components

In [20], Salton and Buckley present a specification for the main function of a term-weighting system. Generally, a typical term-weighting formula is defined as being composed of two component triples: $\langle tfc_q, cfc_q, nc_q \rangle$, which represents the weight of a term in a user query q, and $\langle tfc_d, cfc_d, nc_d \rangle$, which represents the weight of a term in a document d. The term frequency component (tfc) represents how many times a term occurs in a document or query. The collection frequency component (cfc) considers the number of documents in which a term appears. Low frequencies indicate that a term is unusual and thus more important to distinguish documents. Finally, the normalization component (nc) tries to compensate for the differences existing among the document lengths.

Typical term-weighting formulas combine these three components. For instance, as in [20], we can define w_{td}=$tfc_d \times cfc_d \times nc_d$, and w_{tq}=$tfc_q \times cfc_q \times nc_q$, where w_{td} is the weight of term t in document d and w_{tq} is the weight of term t in query q. A common definition for some of these components is tfc_d as the raw term frequency of term t in document d, cfc_d as the inverse document frequency (*idf*) of term t (usually defined as $cfc_d = \log(N/n_t)$, where N is the total number of documents and n_t is the number of documents where term t occurs), and nc_d as the inverse of the size of document d. This is usually called a *TF-IDF* (term frequency–inverse document frequency) weighting scheme.

We can express a ranking function based on such a term-weighting system as follows:

$$sim(q,d) = \sum_{t \in q} w_{td} \times w_{tq} \quad (1)$$

where $sim(q,d)$ is the similarity measure between a query q and a document d.

Ten years after Salton and Buckley's proposal, the work of Zobel and Moffat [26] explored this taxonomy further by adding eight different types of weighting functions. Their approach leads to more than 1,500,000 combinations for calculating the similarity between documents and queries, demonstrating that the space of possibilities for customizing and refining ranking functions is extremely large. This has stimulated the application of effective search space exploration techniques, such as GP [11], for discovering collection-adapted similarity functions.

2.2 Genetic Programming

Genetic Programming (GP), an inductive learning technique introduced by Koza in [11] as an extension to Genetic Algorithms (GA), is a problem-solving system inspired by the idea of *Natural Selection*. The search space of a problem, i.e., the space of all possible solutions to the problem, is investigated using a set of optimization techniques that imitate the theory of evolution, combining natural selection and genetic operations to provide a way to search for the fittest solution.

The evolution process starts with an initial population composed by a set of *individuals*. Generally, the initial population is generated randomly. Each individual denotes a solution to the examined problem and is represented by a tree. To each individual is associated a fitness value. This value is determined by an evaluation function, also known as *fitness function*. The fitness value indicates goodness of an individual and it is used to eliminate from the populations all "unfit" individuals, selecting only those that are closest to the desired goal. The individuals will evolve generation by generation through genetic operations such as *reproduction*, *crossover*, and *mutation*. The reproduction operator simply breeds a new individual. The mutation operator simulates the deviations that take place in the reproduction process. Finally, the crossover operator generates new individuals by the composite of some characteristics present in two other individuals (the *parents*).

Thus, for each generation, after the genetic operations are applied, a new population replaces the current one. The fitness value is measured for each new individual, and the process is repeated over many generations until the termination criterion has been satisfied. This criterion can be a preestablished maximum number of generations or some additional problem-specific success predicate to be reached (e.g., an intended value of fitness for a specific individual).

3. COMBINED COMPONENT APPROACH

Our *Combined Component Approach* (CCA) is a GP-based approach for discovering good ranking formulas. Our goal is to discover new ranking functions more adapted to the specificities of a particular collection. As mentioned before, differently from other GP approaches, our idea consists in examining important information retrieval ranking formulas from systems such as [1,4,18] and extracting from their ranking schemes components such as those described in Section 2.1. These components can be entire formulas or some parts of a ranking formula. Once they are identified, we can use them as building blocks of our GP approach and combine them to generate new ranking functions.

3.1 CCA Modeling for Ranking Function Discovery using Genetic Programming

As in Fan et al. [7], we use a tree data structure to represent a term weighting formula. This tree-based representation allows for easy parsing, implementation and interpretation. Figure 1 illustrates an example individual representing

a TF-IDF formula. The leaf nodes in such trees are called *terminals*, and represent the basic information units that will be composed to create the final formula. Terminals are combined through functions which are represented in the internal nodes. In previous works [7–10, 22], terminals always reflect basic statistics directly derived from the collection, such as term frequency or document size. Our work differs from this approach, as explained below.

Figure 2 displays another individual representing the TF-IDF weighting scheme. In this case, *idf* information is itself a terminal and not the result of a combination of terminals. In previous approaches, this information is a subtree that must be explicitly discovered by the GP evolutionary process. Thus, our CCA approach takes advantage of using information previously known to be effective in other ranking formulas, such as *idf*, pivoted normalization and others, allowing for a more effectively oriented exploration of the search space.

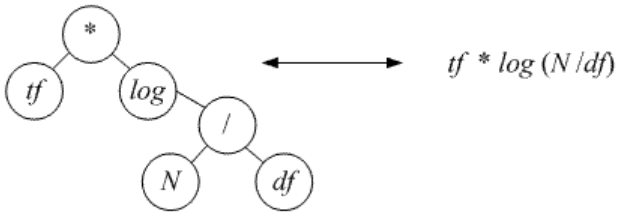

Figure 1: A sample tree for a TF-IDF individual based on statistical information

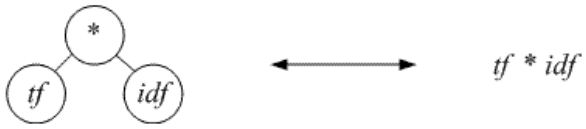

Figure 2: A sample tree for a TF-IDF individual based on our Combined Component Approach

GP Framework

The GP framework is basically an iterative process with two phases: training and validation. For each phase, we select a set of queries and documents from the collection, which we call the *training set* and the *validation set*.

The framework starts with the creation of an initial random population of individuals that evolves generation by generation using genetic operations (reproduction, crossover, and mutation). The process continues until a stopping criterion is met. In the training phase, each time a new generation is created, the fitness function is applied to each new individual, to select only the fittest. Since each individual represents a weighting scheme, applying this fitness function corresponds to ranking the set of training documents according to the set of training queries, using the individual's weighting scheme. The obtained fitness value is simply a quality assessment of the generated ranking.

After the last generation is created, to avoid selecting individuals that work well in the training set but do not generalize for different queries/documents (the *overfitting* problem), the validation phase is applied. In this case, the fitness function is used but on the validation set of queries and documents. Only the individuals that perform the best in this phase are selected as the final solutions. This process is described in Listing 1.

Listing 1: GP Framework used by CCA

```
1  Let 𝒯 be a training set of queries;
2  Let 𝒱 be a validation set of queries;
3  Let N_g be the number of generations;
4  Let N_b be the number of best individuals;
5  𝒫 ← Initial random population of individuals;
6  ℬ_t ← ∅;
7  For each generation g of N_g generations do {
8      ℱ_t ← ∅;
9      For each individual i ∈ 𝒫 do
10         ℱ_t ← ℱ_t ∪ {g, i, fitness(i, 𝒯)};
11     ℬ_t ← ℬ_t ∪ getBestIndividuals(N_b, ℱ_t);
12     𝒫 ← applyGeneticOperations(𝒫, ℱ_t, ℬ_t, g);
13 }
14 ℬ_v ← ∅;
15 For each individual i ∈ ℬ_t do
16     ℬ_v ← ℬ_v ∪ {i, fitness(i, 𝒱)};
17 BestIndividual ← applySelectionMethod(ℬ_t, ℬ_v);
```

Terminals and Functions

An individual is represented by terminals and functions, organized in a tree structure, as shown in Figure 2. Terminals contain information extracted from the main ranking formulas published on IR literature. Table 1 describes all the terminals to be used. Besides these, we also use constant values in the range [0..100]. As functions, we use addition (+), multiplication (∗), division (/) and logarithm (log).

Genetic Operators

Based on Koza [11], we use the genetic operators of reproduction, crossover and mutation as detailed in Section 2.2.

Fitness Function

Since our individuals represent term-weighting schemes to be used in a document ranking function, our fitness function must measure the quality of the ranking generated by a given individual. We experimented with two different functions: (1) non-interpolated average precision over all relevant documents (PAVG) as described in [5, 24], and (2) function FFP4 as defined in [5], a utility function based on the idea that the utility of a relevant document decreases with its ranking order.

Selection of the Best Individual

As mentioned before, we use a validation set to help in choosing good solutions that are not over-specialized for the training queries, i.e. that are able to generalize for unseen queries. In [12], the choice of the best individual is accomplished by considering the average performance of an individual in both the training and validation sets minus the standard deviation value. We call this method AVG_σ. The individual with the highest value of AVG_σ will be selected as the best.

Despite being a balanced approach, our experiments have shown that this method may not lead to the best performance in some runs. We therefore propose a similar method, which also considers the dispersal between training and validation values, but uses the sum of these values in place of

Table 1: Terminals used by our CCA approach in the GP framework

Id	Terminal	Description
t_{01}	tf	Raw term frequency (number of times a term occurs in a document) [20]
t_{02}	$1 + \log(tf)$	Natural logarithm of term frequency factor as presented in [25] to smooth the influence of term frequency
t_{03}	$0.5 + \frac{0.5+tf}{\max tf}$	Term-frequency factor normalized by maximum tf in a document, and further normalized to lie between 0.5 and 1.0 [2, 20]
t_{04}	$\frac{1+\log(tf)}{1+\log(\text{avg}tf)}$	Term-frequency factor normalized by average tf in a document as defined in [21]. Part of SMART weighting scheme formula as defined in [4]
t_{05}	$\frac{(k_1+1)tf}{(k_1((1-b)+b\times dl/\text{avg}dl)+dl)+tf}$	Part of Okapi BM25 ranking formula with term frequency *tfc* and normalization *nc* components [17]
t_{06}	$\log\left(\frac{N}{df}\right)$	Inverse document frequency (*idf*) [20]
t_{07}	$\log\left(\frac{N}{df}+1\right)$	An alternative to inverse document frequency (*idf*) as presented in [25]
t_{08}	$\log\left(\frac{N-df+0.5}{0.5}\right)$	A variation of the Robertson-Sparck Jones weight [16]
t_{09}	$w^{(1)} = \log\left(\frac{N-df+0.5}{df+0.5}\right)$	Robertson-Sparck Jones weight [16–18]
t_{10}	$\log\left(\frac{N-df}{df}\right)$	A probabilistic inverse collection frequency [16, 20]
t_{11}	$\frac{\log\frac{N+0.5}{df}}{\log N+1}$	Part of INQUERY belief function to calculate the belief in a term within a document [1]
t_{12}	$\frac{1}{\sqrt{\sum_{i=0}^{2} w_{i,j}^2}}$	Cosine normalization where $w_{i,j}^2$ is $t_{01} \times t_{07}$ [20, 25]
t_{13}	$\frac{1}{\sqrt{\sum_{i=0}^{2} w_{i,j}^2}}$	Cosine normalization where $w_{i,j}^2$ is $t_{02} \times t_{07}$ [20, 25]
t_{14}	dl	Document length (in bytes) normalization. A normalization technique used in degraded text collections (e.g. OCR-based repositories) [21]
t_{15}	$\frac{1}{(1-slope)+slope\times\frac{\text{avg}t_{13}}{t_{13}}}$	Pivoted cosine normalization with t_{13} terminal as defined in [21]
t_{16}	$\frac{1}{(1-slope)\times\text{avg}dl+slope\times dl}$	Pivoted byte size normalization as defined in [21]
t_{17}	$\frac{1}{(1-slope)\times pivot+slope\times\#\text{ of unique terms}}$	Pivoted unique normalization [21], where *pivot* is the average of the number of unique terms in a document
t_{18}	$\frac{1}{\left(k_1\times(1-b)+b\times\frac{dl}{\text{avg}dl}\right)+tf}$	Normalization factor present in Okapi BM25 scheme, as defined in [17, 18]
t_{19}	$\frac{(k_3+1)\times qtf}{k_3+qtf}$	Query factor present in Okapi BM25 scheme, as defined in [17, 18]
t_{20}	$0.5 + \frac{0.5+qtf}{\max qtf}$	Augmented normalized query term frequency, as defined in [2, 20]

the average. We call this method SUM_σ. More formally, let t_i be the training performance of an individual i, let v_i be the validation performance of this individual, and let σ_i be the corresponding standard deviation. The best individual is selected by:

$$\underset{i}{argmax}((t_i + v_i) - \sigma_i) \quad (2)$$

The experiments described in this paper in Section 4 were performed with both AVG_σ and SUM_σ methods.

4. EXPERIMENTS

In this section we describe our experiments and present the results obtained.

4.1 Data Sets

In our experiments, we use the TREC-8 [24] and WBR99 collections to evaluate our approach. The TREC-8 collection has roughly 528K documents, and 737K distinct terms. Our experiments were performed using 50 topics numbered from 401 to 450. For each topic, there is a set of relevant documents that can be used for testing a new ranking formula. The queries were automatically generated using the title, description and narrative of each topic. We divide the queries into three groups: topics 401-420 were used for the training phase, topics 421-430 were used for the validation phase, and topics 431-450 were used for the test phase.

The WBR99 collection[1], which has been previously used in works such as [15], contains a database of Web pages, a set of queries and a set of relevant documents associated with each query. The relevant documents were generated through a manual evaluation of ranked documents retrieved by a query processor based on the vector space model. The collection has almost 6M Web pages, crawled from the Brazilian web (the *.br* domain), and almost 2.7M distinct terms. It represents a considerably connected snapshot of the Brazilian Web community, which is probably as diverse in content and link structure as the entire Web. Thus, we believe it makes a realistic testbed for our experiments. Queries 1-20

[1]Available at http://www.linguateca.pt/Repositorio/WBR-99/

were used for the training phase, queries 21-30 for validation, and queries 31-50, except query 35, for the test phase. All the results reported in Section 4.4 for both collections are based on the test queries.

4.2 GP Parameters and Setup

Almost all the parameters and configurations were tuned based on Fan et al.'s work [9]. An initial population of 200 individuals was created randomly using the *ramped half-and-half method*. We experimented with PAVG and FFP4 fitness functions, as described in Section 3.1, for the TREC-8 and WBR99 collections. PAVG was more effective for the TREC-8 collection, whereas FFP4 was the best choice for WBR99. Thus, we report only the results obtained with PAVG for TREC-8 and with FFP4 for WBR99. Due to the stability of the results after 30 generations, we defined this value as the termination criterion. We used crossover, reproduction and mutation rates of 90%, 5%, and 5%, respectively. The random seed used was 1234567890. At the end of each generation, the validation phase was run for the top 20 best individuals discovered on the training phase of that generation.

The terminals used were those defined by our CCA approach in Table 1 plus real constant numbers generated randomly. We used the addition (+), multiplication (*), division (/) and protected logarithm[2] (log) functions to combine terminals and subtrees of an individual.

The maximum depth of the generated trees ranged from 3 to 12. We experimented with each value of maximum depth to analyze its influence on the quality of the discovered ranking functions.

4.3 Evaluation and Baselines

We compared the retrieval results of our approach with three others: (i) Okapi BM25, as described in [17] (using parameters k_1=1.2, k_2=0, k_3=1000, b=0.75), (ii) vector space model with standard TF-IDF weighting scheme, and (iii) the GP approach proposed by Fan et al. in [9]. Due to space restrictions, we report only the two best methods per collection — BM25 and FAN-GP for TREC-8, and TF-IDF and FAN-GP for WBR99. TF-IDF outperformed BM25 for the WBR99 collection, probably due to the way that the relevance information was obtained. Despite the fact that Fan et al.'s original work had not varied the tree depth, we allowed the depth to vary between 3 and 12, as in our approach, in order to ensure a fair comparison. The remaining GP parameters were set as defined in that work, except the mutation factor, which was set to 5%. Although Fan et al. ran the experiments without mutation, our experiments showed that this operation leads to better results. We chose the best ranking function discovered by Fan et al.'s approach (henceforth named FAN-GP) to use as a baseline. This selection of the best ranking function was done as described in Section 3.1, due to the AVG_σ and SUM_σ selection methods leading to ranking functions at least as good as the best validation individuals.

To evaluate the performance of our approach against the baselines, we used, as in [24], the (non-interpolated) average precision measure over all relevant documents, the precision at 9 document cutoff values, and R-precision. We also plot all retrieval results in precision-recall curves.

[2]The protected logarithm function, used to prevent numeric overflow, returns zero if its argument is zero and otherwise returns the natural logarithm of the absolute value of its argument [11].

4.4 Results

In this section we present the results of our experiments. The best results were obtained by ranking functions discovered using the SUM_σ method to select the best individuals, and maximum tree depths of 5 and 8 for the TREC-8 and WBR99 collections, respectively. We report the best individuals for TREC-8 considering their global performance. For WBR99, we considered the best performance overall and in the top of the ranking, an important aspect for a Web collection[3]. The best ranking functions discovered by CCA for the TREC-8 and WBR99 collections are shown in Figures 3 and Figure 4, respectively.

```
(*
     (* (log t08) (+ t05 t07))
     (+ (+ (* (+ t19 t05) (+ t07 t06))
           (* (+ t06 t02) (* t16 t18)))
        (/ t07 t19))
)
```

Figure 3: CCA discovered ranking function for the TREC-8 collection

```
(+ (+ (+ 99.09 t11)
   (+ (* (* t07 t10)
      (* t05 (* (+ (* t07 t10) (+ t08 t10)) (* t12 t01))))
      (* (* t07 t10)
      (* t05 (* (+ (* t02 t04) (+ t08 t10)) (* t12 t01))))))
   (+ (* t12 t01)
      (* (* t07 t10)
      (* t05 (* (+ (/ t08 t20) (+ t08 t10)) (* t12 t01)))))
)
```

Figure 4: CCA discovered ranking function for the WBR99 collection

Each discovered ranking function had one terminal related to term frequency component for the queries — t_{19} or t_{20} — and at least three different terminals related to collection frequency — t_{06}, t_{07}, t_{08}, t_{09}, t_{10}, or t_{11}. Figure 3 shows parts of the BM25 ranking function, t_{05}, t_{18}, and t_{19}. Figure 4 presents general TF-IDF components such as t_{01}, t_{02}, t_{07}, and t_{12}. This shows that CCA was able to reuse good components of the best baseline function for each collection and to combine them with others to generate better ranking functions.

Figure 5 displays the evolution process after 30 generations. For each generation, we evaluate the best 20 individuals sorted according to their performance, calculated through the fitness function. As we can see, CCA ranking functions converge faster than FAN-GP. The figures also show that the CCA curves behave very similarly. Despite the fact that training, validation, and test sets present different fitness values, validation and test curves tend to follow the training behavior. In the FAN-GP curve, test and validation curves do not follow the training behavior, which suggests overfitting. The gap between the training curve and the others also indicates overfitting — the greater the gap, the higher the overfitting. As we can see, the CCA curves are also closer than FAN-GP.

[3]In fact, contrary to TREC-8, the best global individual in WBR99 was not the best on the top of the ranking.

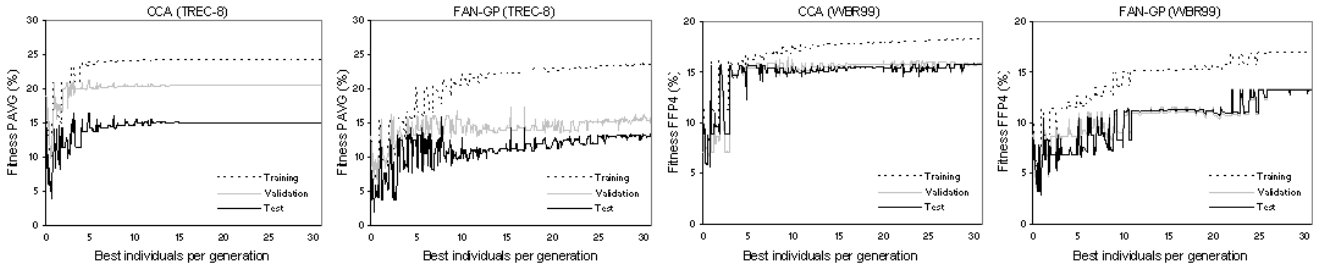

Figure 5: TREC-8 and WBR99 evolution processes for the best 20 individuals in 30 generations

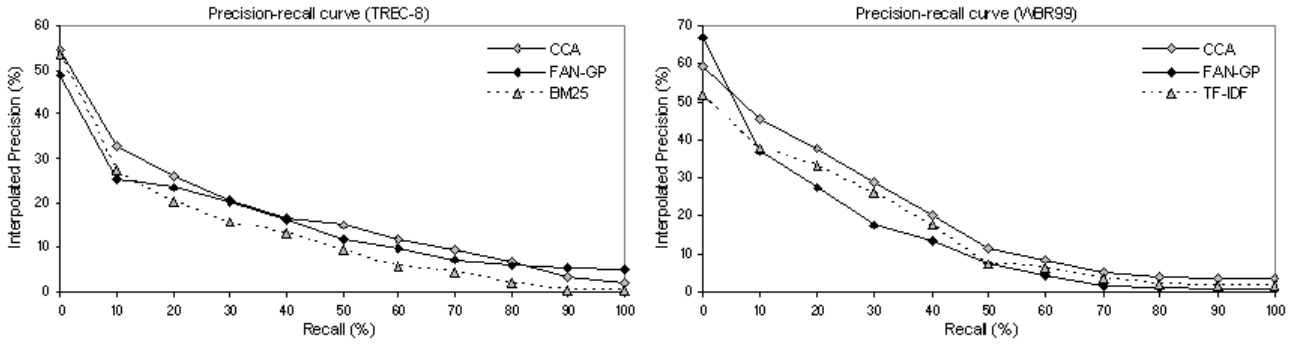

Figure 6: TREC-8 and WBR99 precision-recall curves

Figure 6 shows the 11-point average precision figures for CCA and the baselines on TREC-8 and WBR99 collections. We notice that CCA yields better precision values than TF-IDF, BM25 and FAN-GP throughout almost all recall levels.

Table 2 presents detailed average precision figures at different ranking levels. As we can see, CCA yields better precision than the BM25 and FAN-GP approaches for TREC-8 and better than TF-IDF and FAN-GP for the WBR99 collection. For the TREC-8 collection, CCA reached an average precision of 16.40%, corresponding to gains of 40.87% over BM25 and 14.00% over FAN-GP. For the WBR99 collection, CCA achieved an average precision of 16.68%, corresponding to gains of 21.67% over TF-IDF and 24.85% over FAN-GP. We also observe that FAN-GP did not surpass the TF-IDF results for the WBR99 collection. All the CCA performance improvements related to average precision were statistically significant at $p < 0.1$ for TREC-8 and $p < 0.05$ for WBR99. Table 2 shows the confidence levels (1-p) obtained by the pair-wise t-tests.

We also observe in Table 2 that, for the top 5 documents, CCA improves BM25 and FAN-GP by 18.52% on TREC-8, and improves TF-IDF and FAN-GP by 59.10% and 2.94% on WBR99. Our CCA approach also attains improvements in R-precision of more than 26% over BM25 and 15% over FAN-GP for TREC-8, and more than 8% over TF-IDF and 12% over FAN-GP for WBR99.

Figure 7 presents the average precision for each query. For the TREC-8 collection, we observe that our CCA results improve the performance over FAN-GP in some topics where it had no gain over BM25, such as topics 438, 443, 445, 449, and 450. Figure 7 also shows that CCA's ranking functions only present worse performance than BM25 and FAN-GP in topics where mean average precision is already very low, such as 432, 435, 437, 440, and 448. For the WBR99 collection, CCA obtains better results than the baselines for the queries 31, 32, 33, 34, 44, 47, 48, 49, and 50. CCA stands next to the best results for the queries 39, 40, and 46. Our approach does not surpass FAN-GP for the queries 37, 39, 41, 42, 43, and 45.

Considering the experiment with the depth that produced our best individual for each collection, more than 30 different individuals discovered by CCA surpassed the baselines for TREC-8 and WBR99, which indicates that CCA is able to find several good ranking functions.

5. RELATED WORK

Different approaches to discover ranking functions based on machine learning techniques, such as genetic algorithms and genetic programming, have been proposed in the literature. Fan et al. [7] proposed a new approach to automatically generate term weighting strategies for different contexts, based on genetic programming (GP). They argue that each specific context demands a different term weighting strategy, that is, a ranking function should be adapted to different document collections and users. In their works [8–10] they have demonstrated that GP has been effective at improving the performance of information retrieval tasks. In [5], they also study the effect of different utility functions, i.e., functions that privilege the retrieval of relevant documents on the top of the ranking, when used as fitness functions. In all these works, terminals are based on basic statistical information of the collection, documents and queries, such as term frequencies (tf), total number of documents, length in bytes of a document, number of unique terms in a document, etc. In [8], Fan et al. compare

Table 2: TREC-8 and WBR99 document level average figures for CCA

Level	TREC-8					WBR99				
	Baselines		CCA			Baselines		CCA		
	BM25	FAN-GP	*Precision*	*Gain over BM25*	*Gain over FAN-GP*	TF-IDF	FAN-GP	*Precision*	*Gain over TF-IDF*	*Gain over FAN-GP*
At 5 docs	27.000	27.000	32.000	+18.52%	+18.52%	23.157	35.790	36.842	+59.10%	+2.94%
At 10 docs	29.000	25.500	31.500	+8.62%	+23.53%	26.315	32.632	32.632	+24.00%	0.00%
At 15 docs	25.333	24.667	27.000	+6.58%	+9.46%	30.175	29.474	29.123	-3.49%	-1.19%
At 20 docs	23.750	22.750	25.000	+5.26%	+9.89%	32.105	26.842	27.368	-14.75%	+1.96%
At 30 docs	20.167	21.333	22.167	+9.92%	+3.91%	25.263	21.579	25.965	+2.78%	+20.33%
At 100 docs	15.050	15.150	16.050	+6.64%	+5.94%	13.421	10.737	13.632	+1.57%	+26.96%
At 200 docs	11.900	11.775	12.025	+1.05%	+2.12%	9.000	7.553	8.790	-2.34%	+16.38%
At 500 docs	7.410	7.160	7.650	+3.24%	+6.84%	3.726	3.516	3.737	+0.28%	+6.29%
At 1000 docs	4.910	4.505	5.000	+1.83%	+10.99%	1.968	1.879	2.005	+1.87%	+6.73%
R-precision	16.116	17.703	20.449	+26.89%	+15.51%	20.607	19.830	22.294	+8.19%	+12.43%
Average	11.643	14.388	16.402	+40.87%	+14.00%	13.710	13.361	16.681	+21.67%	+24.85%
Confidence Level				*94.05%*	*98.19%*				*96.96%*	*96.04%*

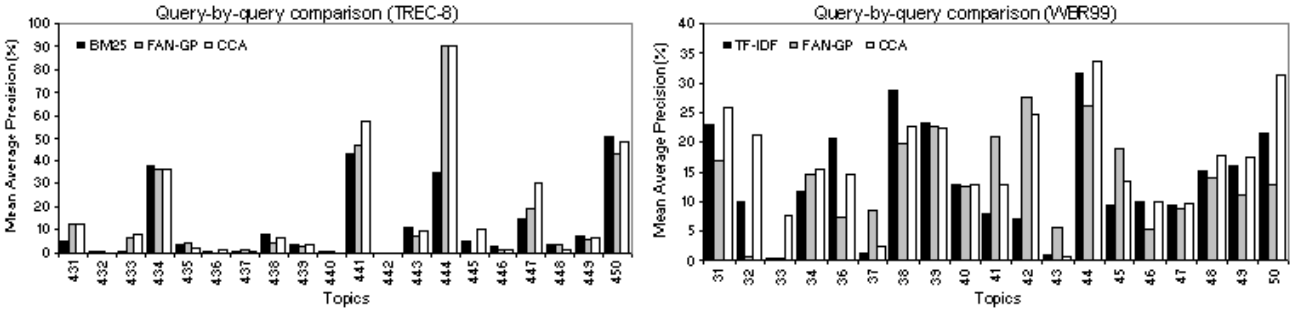

Figure 7: TREC-8 and WBR99 query-by-query comparisons

FAN-GP to a learning approach based on neural networks. Their results showed that FAN-GP overcome substantially the neural network strategy. In [10], FAN-GP exploits structural information of Web documents. Their approach was compared to one based on Support Vector Machines (SVM). FAN-GP manages to improve SVM for both ad hoc and routing tasks in retrieval.

The work in [22] also presents a GP approach, based on statistical information of the collection, documents, and queries. Additionally, and unlike the works of Fan et al., this work adds the baseline functions, such as those in [18,20,21] as individuals in the initial population. According to the author, this addition guarantees that the worst possible performance during the training phase is at least as good as the best baseline of the functions added. The fact that Trotman has chosen to represent individuals (including the baselines) with simple statistical information means that he could not guarantee the integrity of the baseline components, since these could be very distorted or completely destroyed by the evolutionary process. In fact, the best individuals reported by Trotman did not contain components of the baselines. CCA instead guarantees the preservation of these good building blocks and the knowledge they carry, having (indirectly) outperformed Trotman's approach. When applied to a specific collection (FBIS), Trotman was able to achieve an improvement of 32.90% in MAP. This, however, was described as an atypical result. The average gain for all tested collections in Trotman's best run was around 8%, with negative gains for some collections. Our results surpassed BM25 by 40.87% for TREC-8, suggesting a substantial improvement over Trotman's approach. Finally, different from CCA, in which we advocate a collection-based approach, Trotman has the goal of finding ranking functions good for all colections.

In [13], Oren explores several tf-idf functions generated by a GP approach. His work uses statistical information of the collection, idf, and two instances of tf as terminals. However, his improvements over a basic tf-idf strategy were not significant.

In contrast to all of these aforementioned works, our approach uses parts of well-known, significant, and proven effective ranking formulas as terminals, for representing term-weighting components, instead of simple statistical information. As mentioned before, our hypothesis is that providing richer components for GP to work with will allow the discovery of better final ranking formulas. In fact, our approach proved to be more stable and consistent in generating effective ranking functions than previous work, which used only basic statistical information.

Other works that are also somewhat related to ours have focused on the combination of results from different ranking formulas. In [3], Bartell et al. show that retrieval effectiveness can be improved significantly by a combination of the results of a number of different retrieval algorithms (or *experts*), since these emphasize different aspects of the documents to be retrieved.

Pathak et al. [14] introduces a method of utilizing Genetic Algorithms in Information Retrieval tasks. Specifically, it shows how GA can be used to adapt several matching functions that are used to match documents descriptions with query descriptions. Vogt et al. [23] propose a linear combination model for fusion of ranking results. This model

combines the result lists of multiple IR systems by scoring each document using a weighted sum of the scores from each of the component systems. The work in [6] also uses Genetic Algorithms to combine different expert matching functions. The weights associated with combinations are evolved using GA. Our approach differs from these, since we do not combine ranking formulas or result lists, but components or portions of ranking functions from different retrieval systems to generate a completely new ranking formula.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced a new approach based on the combination of term weighting components, extracted from well-known information retrieval ranking formulas, using genetic programming. We show that our *Combined Component Approach* (CCA) improves the retrieval performance compared to standard TF-IDF, BM25 and other GP-based approaches [9, 22] that use only basic statistical information derived from collections and documents. We used the TREC-8 and WBR99 collections to validate our approach.

Using the TREC-8 collection, our experiments showed that CCA leads to significant improvements in retrieval effectiveness. Mean average precision was improved by 40.87% and 14.00% over BM25 and FAN-GP respectively. R-precision and precision at the top of the ranking improvements were also observed. For the WBR99 collection, we obtained improvements of 21.67% and 24.85% in mean average precision over TF-IDF and FAN-GP, respectively.

Examining the CCA evolution process, we observed that CCA ranking functions converged faster than FAN-GP. Our approach also reduced overfitting. Results obtained by CCA lead us to conclude that the use of meaningful terminals instead of simple statistical information improves the quality of the process of discovering ranking functions with GP.

For future work, we will investigate ranking formulas from other IR models such as the Set-based Model [15] to extract new terminals. We will utilize the structural information within documents to compare our approach to others, such as [10], for Web search. We also will investigate the influence of the mutation factor, tree depth, and population length on the discovery of good ranking functions, considering the use of meaningful terminals and statistical information. Finally, but not less important, we also intend to examine closely the discovered best ranking functions to understand better how they work and the reasons for their effectiveness.

7. ACKNOWLEDGMENTS

Special thanks go to Prof. Edward Fox for his very helpful comments. This work is partially supported by projects GERINDO (CNPq/CT-INFO 552.087 /02-5), 5S-VQ (CNPq /CT-INFO 55.1013/2005-2), FCT project ref. POSC/EIA/ 58194/2004, GRICES/CNPq bilateral cooperation project "ADAPTINF", by grant 10023 - UFMG/RTR/PRPQ/RECEM-DOUTORES/04 (sub-grant 32 - PROGRAMACAO GENETICA), and by an individual grant from CNPq to Marcos André Gonçalves.

8. REFERENCES

[1] J. Allan, J. P. Callan, F. Feng, and D. Malin. INQUERY and TREC-8. In *Proceedings of TREC-8*, pages 637–644, Gaithersburg, MD, 1999. NIST Special Publication 500-246.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman, Boston, MA, 1999.

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th ACM SIGIR*, pages 173–181, 1994.

[4] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using smart: TREC 4. In *Proceedings of TREC-4*, pages 25–48, Gaithersburg, MD, 1996. NIST Special Publication 500-236.

[5] W. Fan, E. A. Fox, P. Pathak, and H. Wu. 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):628–636, 2004.

[6] W. Fan, M. Gordon, and P. Pathak. On linear mixture of expert approaches to information retrieval. *Decision Support Systems*, 42(2):975–987, 2006.

[7] W. Fan, M. D. Gordon, and P. Pathak. Personalization of search engine services for effective retrieval and knowledge management. In *Proceedings of the 21st Intern. Conf. on Inf. Systems*, pages 20–34, Brisbane, Australia, 2000.

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

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

[10] W. Fan, M. D. Gordon, and P. Pathak. Genetic programming-based discovery of ranking functions for effective web search. *Journal of Manag. Inf. Syst.*, 21(4):37–56, 2005.

[11] J. R. Koza. *Genetic Programming: On the programming of computers by natural selection*. MIT Press, Cambridge, 1992.

[12] A. Lacerda, M. Cristo, M. A. Goncalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *Proceedings of the 29th ACM SIGIR*, pages 549-556, 2006.

[13] N. Oren. Reexamining tf.idf based information retrieval with genetic programming. In *Proceedings of the SAICSIT 2002 Conference*, pages 224–234, 2002.

[14] P. Pathak, M. Gordon, and W. Fan. Effective information retrieval using genetic algorithms based matching functions adaptation. In *Proceedings of the 33rd HICSS*, Hawaii, 2000.

[15] B. Pôssas, N. Ziviani, J. Wagner Meira, and B. Ribeiro-Neto. Set-based vector model: An efficient approach for correlation-based ranking. *ACM TOIS*, 23(4):397–429, 2005.

[16] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[17] S. E. Robertson and S. Walker. Okapi/keenbow at TREC-8. In *Proceedings of TREC-8*, pages 151–162, Gaithersburg, MD, 1999. NIST Special Publication 500-246.

[18] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of TREC-3*, pages 109–126, Gaithersburg, MD, 1995. NIST Special Publication 500-226.

[19] G. Salton. *The SMART retrieval system - Experiments in automatic document processing*. Prentice Hall Inc., Upper Saddle River, NJ, 1971.

[20] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[21] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM SIGIR*, pages 21–29, 1996.

[22] A. Trotman. Learning to rank. *Information Retrieval*, 8(3):359–381, 2005.

[23] C. C. Vogt and G. W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[24] E. M. Voorhees and D. Harman. Overview of the eighth Text REtrieval Conference (TREC-8). In *Proceedings of TREC-8*, pages 1–24, Gaithersburg, MD, 1999. NIST Spec.Publ. 500-246.

[25] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[26] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1):453–490, 1998.