Proceedings of the Fourth IASTED International Conference
COMPUTATIONAL INTELLIGENCE
July 4-6, 2005, Calgary, Alberta, Canada

A SEARCH ENGINE THAT LEARNS

Jeffrey K Elser, John Paxton
Montana State University - Bozeman
Computer Science Department
Bozeman, MT USA
elser@cs.montana.edu, paxton@cs.montana.edu

ABSTRACT

Tuning a local Web site to generate better local search results is a time consuming and tedious process. In this paper, we describe a technique that can help to automate this process. Specifically, when a genetic algorithm is applied to a local search engine's parameters, the performance of the local search engine can be improved. Once good values for the search engine have been learned, it is easy to identify local Web pages that are candidates for further improvement.

KEY WORDS

Genetic Algorithms, Search Engine Enhancement

1. Introduction

The Customer Relationship Management (CRM) software developed by RightNow Technologies includes many features designed to streamline customer interaction. One function of RightNow's CRM suite is a knowledge base where customers may ask questions that are answered by the software product rather than by company employees. The knowledge base works through spidering and indexing of a company's Web site, allowing pre-existing information to be utilized. [1, 2, 3]

A fairly standard approach for spidering and indexing the Web site is used. When a company installs RightNow's software, a service tech performs the configuration and spiders the company's Web site. Initially, a list is built containing the location of words found on the page. These words can be found in the normal text body, HTML tags (e.g. H1), meta-tags and even URLs. Finally an index is created from this list using the software's system of weighting.

Unfortunately, a company's Web pages are often not optimized for local searching. For example a company might place its name in the description meta-tag of every page on its Web site in order to rank its pages higher than its competitors' pages on Internet-wide search engine rankings. However, this homogeneous local structure might reduce a local search engine's ability to differentiate between the content of local pages.

Because a company is in control of both its Web site and its local search engine, it should be possible to have its local search engine perform more accurately. As it may be very labor intensive for the company to manually update and optimize each Web page, a more reasonable request is to ask the local webmaster to alter a small subset of the Web site based on explicit given instructions.

This problem can be approached in two ways. First, the local search engine can be tuned using an automated process. Second, changes to the Web site that cannot be made through automation can then be suggested. The focus of this paper is on the first of these two approaches. The remainder of this paper is organized as follows. Section 2 presents relevant background on search engines and genetic algorithms. Section 3 describes a method for automatically optimizing a local Web site. Section 4 presents some preliminary results. Section 5 summarizes the findings and presents some ideas for future research.

2. Background

2.1 Search Engine Background

The search engine used in RightNow's CRM product spiders and indexes a website through common spidering methods. The important feature of this search engine is the system of weights used to create the index. These weights act as multiplying factors when the score of a search term's occurrence in a document is calculated. For example, if the search term is found in the title section of a Web page, and the title weight is 100, then the score of that term is multiplied by 100 and added to the document's score. The score for each document is thus the sum of the search term occurrences multiplied by their weights. The result of a search lists the pages based on their final scores in decreasing order. The first column in Table 1 contains a list of some of the standard weights and the second column contains their default, pre-optimized values. The third column of Table 1 will be discussed later.

A majority of the weight identifiers in Table 1 refer to simple HTML or Meta tags such as <title> or <h1>.

However the meaning of a few of the tags may not be obvious and their functions will be described below.

The meta-description identifier refers to a search engine specific Meta tag. This tag is only understandable to RightNow Technologies' search engine. It is not considered by most Internet-wide search engines and does not affect the page's appearance in a browser.

The multi-match weight is applied when more than one keyword occurs in a search. For example, if the search terms are "rocking" and "chair" and both are found in a document, the multi-match weight is applied to the document. This weight has no effect when the Boolean operator joining the two words is AND instead of OR.

The backlink weight is applied as a multiplier to the ratio of the number of links coming into a page versus the number of links going out. A page that has many outgoing links (such as a table of contents) will have its score reduced. A page that has many incoming links (such as one that contains important information) will have its score enhanced. When a Web site has a common tree-like structure, a high backlink weight causes leaf node pages to be boosted in the returned results.

Weight Identifier	Default Value	First Test Case Results
backlink	1000.0	510.0
description	150.0	980.0
keywords	100.0	66.0
title	100.0	180.0
meta-description	50.0	920.0
heading 1	5.0	130.0
heading 2	4.0	340.0
heading 3	3.0	640.0
heading 4	1.0	720.0
heading 5	1.0	430.0
author	1.0	440.0
multi-match	1.0	170.0
text	1.0	0.0
url text	1.0	540.0
date	0.35	140.0
heading 6	0.0	0.0

Table 1

2.2 Genetic Algorithm Background

A genetic algorithm (GA) is used to optimize the search engine parameters to achieve better rankings within the local Web site automatically. Rather than writing a GA from scratch, GAlib's prewritten functions were utilized [4]. GAlib is a collection of genetic algorithm functions written in C++.

Goldberg's Simple GA [5] is used in the initial experiments. The main elements of the simple GA are standard mutation, standard crossover, elitism, and non-overlapping populations. The fitness function will be explained in Section 3.

3. Approach

To produce better search engine rankings, the feature weights in the search engine's configuration file need to be improved.

Initially the Webmaster must supply training data. He or she must identify the ranked pages that should result from a particular search query. This information is stored in a batch file. It is possible that user input can be used to create this batch file dynamically [6, 7]. This is discussed further in the conclusions and future directions section.

The Web sites used for this experiment were created from 11 newsgroup articles. These articles were selected from a 20,000 article data set hosted on the UCI Knowledge Discovery in Databases Archive [8].

The articles were chosen for their structure and word count. Larger documents tend to work better in a search engine simply because they are more likely to contain multiple instances of the search term. The 11 selected articles were formatted by hand to include HTML formatting tags and relevant Meta tags.

The GA begins by creating a random population of genomes. Two different population sizes, 1000 and 10000 were tested but the results were almost identical. All the data in this paper was collected from tests using populations of size 1000. For the problem at hand, the genome contains 16 real numbers ranging from 0.0 to 1000.0. Each number corresponds to one of the weights shown in Table 1. Random initial values is a commonly used technique for reducing the number of generations required for a genetic algorithm to converge upon an acceptable result. There is evidence that the quality of the random number generator affects performance as well, but for these experiments, only GAlib's built in random number generator was used [9, 10, 11]. In the future, experiments will be run using different random number generators.

The GA executes for a predetermined number of generations. Elitism is turned on to ensure that the fittest individual is retained from one generation to the next. The probability of mutation is set to 0.01 and the probability of crossover is set to 0.6.

A fitness function is required for all GAs. The fitness function for this problem is a distance measure between the top ten actual rankings of the Web pages and the top ten desired rankings from the batch file (if there are that many). Equation 1 defines the fitness function where D is the absolute value of the difference between a page's actual ranking and its desired ranking. For example if a page's actual ranking is 5 and its desired ranking is 2,

then the distance is 3. In the batch file, if there are fewer than 10 desired rankings, then the unspecified positions are considered to match perfectly.

$$F = \frac{1}{\sum D + 101} \qquad \text{Equation (1)}$$

There are two special cases for the fitness function. First, if the page is not included in the top ten results in the actual rankings, but it should be according to the desired rankings, then D is set to 100 for that page. Second, if the actual ranking matches the desired ranking, then D is set to -10 for that page. This defines the largest possible distance to be 1000 and the smallest distance as -100. Substituting those values into the fitness function, the largest distance gives a fitness value of 1/1101 or 0.0009, and the smallest distance gives a perfect fitness value of 1.

4. Preliminary Results

Twelve tests were performed on the 11 Web pages. The search query "introduction" was used throughout the tests. Several of the test pages were large introductory texts and FAQs. Each of these articles contained several occurrences of the query term "introduction" in a variety of HTML and Meta tags. The other pages were argument style posts that contained few if any occurrences of the query term.

The 12 tests were designed so that each one has a different desired ranking. The first test's desired ranking was realistic in the sense that a real Webmaster chose it. The desired results for the other 11 tests were picked at random.

The tests performed show that some improvement in ranking is possible, although not guaranteed. In four of the twelve tests, including the realistic one, perfect rankings were achieved. In four other tests, the rankings were improved as the search engine weights were modified by the genetic algorithm. In the remaining four tests, no improvement was attained beyond the initial random weighting. The average distance between the actual and the desired ranking for these last four tests was 5, which as shown in Figure 1, is a common initial distance. This eliminates lucky initial ordering as a factor for lack of improvement.

While 100% accuracy was achieved in some test cases, this level of accuracy was not always possible. The main reason for the GA to fail to achieve perfect rankings lies in not allowing negative weighting. For example, maintaining other factors constant, if one page has three occurrences of a keyword in its body text, while the another page has only one occurrence of the keyword in its body text, then no value for the body text weight will cause the second page to be ranked ahead of the first page. The more pages that appear in the training file, the higher the chances are that there will be no perfect set of search engine weights.

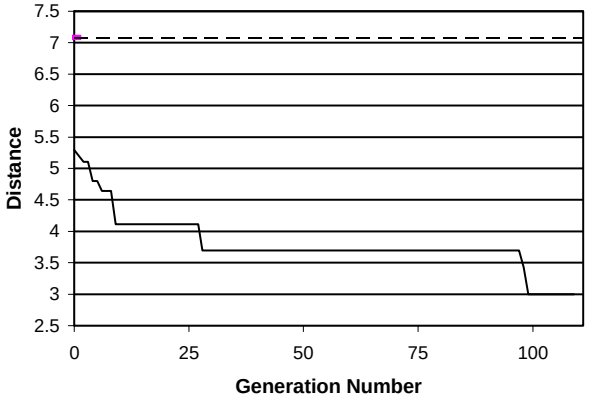

Figure 1: The actual distance between real and desired results plotted over the number of generations.

In Figure 2, the average fitness values from the 12 tests are plotted against the number of generations. It is helpful to review Equation 1 before proceeding. The possible range of values from the fitness function is from 1 to 1/1101. However, it is necessary to realize that after excluding the perfect value of 1, the next best fitness value possible is 1/23 or 0.0435. The score 1/23 comes from 8 perfect matches (d is -10 for each) and the two remaining pages being out of order by just one position (d is 1 for each). For example, if the desired ranking was {5, 2, 1, 3, 4, 6, 7, 8, 9, 0}, and the actual ranking was {5, 2, 3, 1, 4, 6, 7, 8, 9, 0}, then the score would be 1/23. The steps in the graph show where individual tests jump from 1/23 to the optimal solution of 1. By the 200th generation, four out of 12 of the tests achieved the desired ranking. These results might be a feature of the small search space. In the future, larger data sets should be studied.

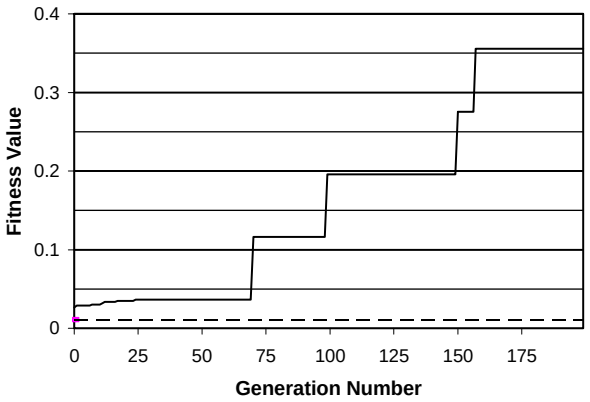

Figure 2: The average fitness values from 12 test cases plotted over the number of generations.

The dashed line in Figure 2 represents the performance using the default weights. As mentioned briefly above, the default weights were chosen based on a combination of intuition and trial and error. Interestingly, using the default weights was not an optimal solution for any of the 12 tests, and the distance from the desired ranking for the one realistic test was 8. While these default weights may make sense when considering which HTML and Meta tags are important, it is apparent that better configurations are often possible.

One possible argument to support these lower results of the default weights is to hypothesize that the web pages being tested are not properly tagged. If this is assumed true, using standard search engine optimization techniques could help the pages rank better. [12, 13] Unfortunately that position meets with a lot of resistance from real life Webmasters. As discussed in the introduction, improving local search results automatically is one main goal of this research. Therefore since these test Web pages are representative ones, they are relevant and valid for this research despite apparent shortcomings in design and content.

The fitness function is not a smooth one (consider the -10 bonus for a perfect match and the +100 penalty for a desired page not being listed in the top ten). While Figure 1 shows the average solution improving over time, it is difficult to see optimization occurring because the fitness values change non-linearly.

In Figure 1, the average actual distance is plotted against the number of generations using a simplified fitness function. The new fitness function does not reward perfect matches, nor does it add a penalty for desired results that do not appear in the top ten actual results.

Using the simplified fitness function, improvement still occurs. However, only one test now achieved the optimal solution. To allow for a better comparison between the original fitness function and the simplified one, consider that the final average distance in Figure 2 (approximated from the final average fitness value) is approximately 2. Comparing that to the final average distance of approximately 3 in Figure 1, it is evident that the simpler function is not able to match the performance of the original equation. However, even the simpler fitness function outperforms the default weights, as depicted by the dashed line in Figure 1.

5. Conclusions and Future Directions

A genetic algorithm is not a magic bullet that can configure a search engine to rank pages perfectly. However, as this work shows, the performance of a local search engine can be improved. The level of improvement attainable remains a future research question.

The fitness function for this GA was not very smooth because of the addition of the perfect result reward of -10 and the missing result penalty of 100. That function performed better than using just the distance and ignoring the two special cases, but it seems likely that a smoother function that includes the special cases would perform even better. Therefore the first area to look into will be developing a smoother fitness function.

Multi-objective genetic algorithms are a possible research path to optimize the search engine weights for larger, more complex page sets [14].

For pages that are still ranked incorrectly after the search engine is optimized by a GA, weights in the configuration file can be used to make further recommendations for improvement. For example, if the description weight is large and the misclassified page has no description, it seems appropriate to recommend writing a description for the page.

Another way to deal with a misclassified page is to automatically generate a value to use in one of the search engine specific meta-tags. This allows artificial content to be added to a page without altering its visible contents. In general, altering the search engine specific meta-tags shouldn't affect the page's ranking with other Internet-wide search engines [15]. Therefore, to deal with problem pages, the program could add artificial content to Web site meta-tags as a temporary solution, and then make recommendations for additional content to be added later by the Webmaster as time allows.

The search engine configuration file weights can also be used to make design change suggestions. For example, if the GA finds that the title tag's weight should be very small, a recommendation could be made to use title tags more accurately. (In general, the title of a web page is considered to be a very important aspect by many Internet-wide search engines.) In this case, the small weight of the title tag likely indicates that the title tags are either written poorly or missing altogether.

Because creating a batch file of desired rankings is tedious work for the Webmaster, it is worthwhile to explore alternate methods of discovering that information. As was mentioned in the approach section, it may be possible to unobtrusively collect information from users about which pages should be ranked higher or lower [6, 7]. When a user conducts a search, statistics can be kept concerning what results the user chooses. The user's final choice could be given special attention since it may be assumed that the user has found the information they were looking for. However, caution is required when using implicit user data because the statistics can be easily misinterpreted. For example, if a search engine returns 10 results per page, it is likely that those 10 results will be clicked more often largely because they are on the first page, and not because they are more relevant.

More research is required in the area of implicit user feedback before it can be relied upon for the initial training data. A more reasonable approach might be to bootstrap the search engine with the batch file created by the Webmaster, and then later refine the search engine by cautiously adding the data from implicit user feedback. This will be an interesting and challenging area to study when this research is put into use by real world search engine and actual implicit feedback can be gathered.

Thus, although some initial progress has been made, much research remains. We are excited to continue exploring this problem.

6. Acknowledgements

We would like to thank RightNow Technologies and the RightNow applied research team for funding the research grant that made this work possible. We also appreciate being provided with access to relevant software.

7. References

1. RightNow Technologies, On Demand Customer Relationship Management Software, http://www.rightnow.com

2. Durbin, S., Warner, D., Richter, N. & Gedeon, Z. Management for Web-Based Customer Service. *Organizational Data Mining: Leveraging Enterprise Data Resources for Optimal Performance. Edited by Nemati and Barko. Idea Group Inc*., 92-108, 2004.

3. Durbin, S., Warner, D., Richter, N. & Gedeon, Z. Information Self-Service with a Knowledge Base that Learns. *AI Magazine, 23*(4), 41-49, Winter 2002.

4. GALIB, A C++ Library of Genetic Algorithm Components, http://lancet.mit.edu/ga

5. Goldberg, D. E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.

6. Qi, H., Hartono, P., Suzuki, K., Hashimoto, S., Sound database retrieved by sound, *Acoustical Science and Technology, Vol. 23*, No. 6, pp. 293-300, 2002.

7. Boyan, J., Freitag, D., Joachims, T., A Machine Learning Architecture for Optimizing Web Search Engines. *Proceedings of the AAAI Workshop on Internet Based Information Systems*, 1996.

8. Hettich, S. and Bay, S. D. The UCI KDD Archive. University of California - Irvine, Department of Information and Computer Science. http://kdd.ics.uci.edu

9. Daida, J., Ross, S., McClain, J., Ampy, D., & Holczer, M. Challenges with Verification, Repeatability, and Meaningful Comparisons in Genetic Programming. *Genetic Programming 97. San Francisco, CA: Morgan Kaufmann Publishers*, 64-69, 1997.

10. Daida, J. M., Ampy, D. S., Ratanasavetavadhana, M., Li, H., & Chaudhri, O. A.. Challenges with Verication, Repeatability, and Meaningful Comparison in Genetic Programming: Gibson's Magic. *Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco, CA: Morgan Kaufmann Publishers*, 1851-1858 (Volume 2), 1999.

11. Meysenburg, M., & Foster, J. Random Generator Quality and GP Performance. *Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco, CA: Morgan Kaufmann Publishers*, 1121-1126 (Volume 2), 1999.

12. Thurow, S., *Search Engine Visibility*, New Riders, 2003.

13. Kent, P., *Search Engine Optimization for Dummies*, Wiley, 2004.

14. Fonseca, C.M., Fleming, P.J., Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. *Genetic algorithms: Proceedings of the Fifth International Conference, Morgan Kaufmann, San Mateo, CA*, 141-153, 1993.

15. Sullivan, D., *How To Use HTML Meta Tags, Search Engine Watch*. http://www.searchenginewatch.com, December 5, 2002.