ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

M. Caramia, G. Felici, A. Pezzoli

IMPROVING SEARCH RESULTS WITH DATA MINING IN A THEMATIC SEARCH ENGINE

R. 574 Settembre 2002

Massimiliano Caramia – Istituto per le Applicazioni del Calcolo IAC-CNR, Viale del Policlinico, 137 -00161 Roma, Italy. Email: caramia@iac.rm.cnr.it.

Giovanni Felici – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email : felici@iasi.cnr.it.

Alessia Pezzoli – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, Viale Manzoni 30 - 00185 Roma, Italy.

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica, CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

The problem of obtaining relevant results in web searching has been tackled with several approaches. Although very effective techniques are currently used by the most popular search engines when no *a priori* knowledge on the user's desires beside the search keywords is available, in different settings it is conceivable to design search methods that operate on a thematic database of web pages that refer to a common body of knowledge or to specific sets of users. We have considered such premises to design and develop a search method that deploys data mining and optimization techniques to provide a more significant and restricted set of pages as the final result of a user search. We adopt a vectorization method based on *search context* and *user profile* to apply clustering techniques that are then refined by a specially designed genetic algorithm. In this paper we describe the method, its implementation, the algorithms applied, and discuss some experiments that has been run on test sets of web pages.

Key words: Search Engines, Web Mining, Clustering, Genetic Algorithms.

1. Introduction

The recent improvements of search engines technologies have made available to the internet users an enormous amount of knowledge, that can be accessed in many different ways. The most popular search engines on the market are now providing search facilities for databases containing billions of web pages, where queries are executed instantly. These extraordinary results have been possible thanks to evoluted indexing systems and very powerful and efficient software and hardware components. Although, many researchers still question whether it is possible to construct search tools that are able to provide more significant results to the users. The focus is thus switching from *quantity* (maintaining and indexing large databases of web pages and quickly select pages matching some criterion) to *quality* (identifying pages with a high quality for the user). Such a trend is motivated by the natural evolution of the internet users, that are now more selective in their choice of the search tool and may be willing to pay the price of providing extra feedback to the system and wait more time in order to have their queries better matched. In this framework, relevant work has been done in [12, 13], and several have considered the use of Data Mining and Optimization techniques, that are often referred to as *Web Mining*.

This paper presents a method for improving standard search results in a thematic search engine, where the documents and the pages made available are restricted to a finite number of topics, as well as the users are considered to belong to a finite number of user profiles. The method uses clustering techniques to identify some structure in the starting set of pages, and then a metaheuristic that tries to optimize the query results based on the results of the clustering.

The use of these techniques is not new in web mining applications. There has been extensive research on how to improve retrieval results by employing clustering techniques. In several studies the strategy was to build a clustering of the entire document collection and then match the query to the cluster centroids (see for example [35]). More recently, clustering has been used for helping the user in browsing a collection of documents [8] or in organizing the results returned by a search engine [25, 41], or by a metasearch engine [39] in response to a user query. In [23] document clustering has also been used to automatically generate hierarchical clusters of documents. A different approach uses already existing document taxonomy as clusters to produce an effective document classifier for unclassified documents [1]. As discussed in [33], the use of clustering in information retrieval (IR) is based mostly on the *cluster hypothesis*: "closely associated documents tend to be relevant to the same request". Several researchers have shown that the cluster hypothesis also holds in a retrieved set of documents [17], but they do not study how the clustering structure may help a user to find relevant results more quickly.

Metaheuristics, and more precisely, genetic algorithms, have been implemented in IR by several researchers and the results indicate that these algorithms could be efficient. In [15] Gordon presents a genetic algorithm based approach to improve document indexing. In that approach, the initial population is represented by a collection of documents judged relevant by a user, which is then evolved through generations and converges to an optimal population with set of keywords which best describe the documents. In [16] the same author adopts a similar approach to document clustering, where a genetic algorithm is used to adapt subject descriptions so that documents become more effective in matching relevant queries.

In [37] the authors apply genetic algorithm in information retrieval in order to improve search queries that produce better results according to users' preferences. In [5] Chen uses genetic algorithms to optimize keywords that were used to suggest relevant documents. Also in [5, 24, 30, 31, 36, 37] several approaches based on genetic algorithms to enhance the query description

are discussed. In [3] a genetic algorithm is deployed to find an optimal set of documents that best match the user's need. The genetic algorithm attempt to involve a population of queries towards those improving results of the information retrieval system. In [19] a method for extracting keywords from documents and assigning them weights is proposed. Both documents and queries are represented as vectors in which the components correspond to significant keywords extracted from documents. A genetic algorithm is then used for assigning weights to the keywords; such weights are then used to form a query vector through which relevant documents may be detected. In [38] a prototype agent for collecting and filtering information from the Internet according to a user profile is described. This intelligent agent includes a genetic algorithm which collects and evaluates new HTML documents from the Web, using information included in examples provided by the user, and returns the user with pages that have high score.

In this paper, yet another way of using genetic algorithms in search engines is proposed. We first use clustering to identify, in the set of pages resulting from a simple query, subsets that are homogeneous with respect to a vectorization based on *context* or *profile*; then we construct a number of small and potentially good subsets of pages, extracting from each cluster the pages with higher scores. Operating on these subsets with a genetic algorithm, we identify the subset that has the property of combining a good overall score with a high internal dissimilarity. This provides the user with few non-duplicated pages that represent more correctly the structure of the initial set of pages.

The outline of the paper is as follows. In Section 2 we provide a general description of the method, illustrating its basic components and its characteristics. Sections 3 to 5 treat in more detail each of the steps that define the method: page vectorization, page clustering and the implementation of the genetic algorithm, respectively. In Section 6 we summarize some experiments that were conducted with the proposed method, based both on randomly generated instances and on real data. Some concluding remarks are given in Section 7.

2. Description of the method

Let P be a set of web pages, and indicate with $p \in P$ a page in that set. Now assume that P is the result of a standard query to a database of pages, and thus represents a set of pages that satisfy some conditions expressed by the user (for example, that satisfy a boolean expression on some search keywords). Each page $p \in P$ is associated with a score, based on the query that generated P, that would determine the order by which the pages are presented to the user who submits the query. The role of this ordering is crucial for the quality of the search: in fact, if the dimension of P is relevant, the probability that the user considers a page p strongly decreases as the position of p in the order increases.

This may lead to two major drawbacks: a) the pages in the first positions may be very similar (or even equal) to each other; b) pages that do not have a very high score but are representative of some aspect of set P may appear in a very low position in the ordering, with a negligible chance of being looked at by the user. Our method tries to overcome both drawbacks, focusing on the determination of a small set of pages, selected from the initial set P, that, beside containing pages with a high score, also are different from each other and are chosen from different regions of some space where the pages are represented.

One of the conditions needed to apply our approach is the availability of additional information from the user, who indicates a search context or a user profile, where:

- the search context is a general topic to which the search is referred to, that is not necessarily

linked with the search keywords that generated the set P; it may be viewed as an item of a topic catalog or directory as the ones that are frequently provided by the last generation of search engines (i.e., catalogs);

- the user profile is a subjective identification of the user, that may be either provided directly by choosing amongst a set of predefined profiles, or extracted from the pages that have been visited more recently by that user.

The basic idea of the method is to use the information conveyed by the search context or the user profile to analyze the structure of set P and determine in it an optimal small subset that better represents all the information available. This is done in three steps, that we sketch below and describe in the following sections.

First, the search context and the user profile are used to extract a finite set of significant words or page characteristics that is

used to create, from all pages in P, a vector of characteristics (page vectorization). Such vectorization represents a particular way of "looking" at the page, specific of each context/profile.

Second, the vectorized pages are analyzed by a *clustering algorithm*, that partitions them into subsets of similar pages. This induces a two-dimensional ordering on the pages, as each page p can now be ordered according to the original score within its cluster. At this point the objective is to provide the user with a reduced list that takes into account the structure identified by the clusters and the original score function. This is done in the third step, where the pages that have higher score in each cluster are selected to compose an initial population that is then analyzed by a *genetic algorithm*. The initial chromosomes are small subsets of pages extracted in turn from the clusters, starting from the pages with higher score; this way the initial population is composed of pages with high score that have a high probability of being different from each other. The genetic algorithm operates using a *fitness function* that combines the score with a measure of the similarity of the pages in the same chromosome. Once the *genetic algorithm* has been run, the best chromosome constitute the subset of pages to be returned to the user.

Figure 1 depicts the overall scheme of how the system works.

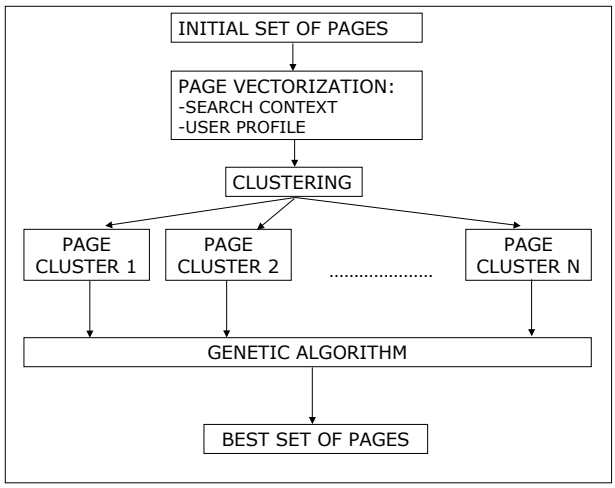

Figure 1: The scheme of the method

The quality of the final subset of pages depends, amongst other things, on the way the genetic algorithm is implemented; that is, how the fitness of each chromosome is measured and how

chromosomes are selected and combined in each iteration. The main idea is that, when properly designed, the genetic algorithm can generally determine chromosomes that are sufficiently heterogeneous and whose pages have good values for the original score. We devote Section 5 to these details.

One may question whether the described method can be run on-line in a search engine as the standard execution of a user's query. We believe that with proper bounds on the parameters and a proper choice of the algorithms deployed, the computational effort can be dealt with and that only few seconds may be added to the overall search process. Moreover, the computation power available on commercial personal computers has and will increase significantly in the future.

Another issue that may rise related to the applicability of this method is that a high level of interaction with the user is required, as a search context has to be selected or be identified into a given profile. But the average skill of the internet user increased very rapidly, together with the need for precise and effective search methods. For this reason users should be willing to provide the additional information needed as the price for better results.

3. Page Vectorization

The first step of the method is the representation of each page that has been acquired by a vector of finite dimension m, where each component represents a measure of some characteristic of the page (*page vectorization*). In the most plain setting, each component of the vector is the number of occurrences of a particular word; one may also consider other measurable characteristics that are not specifically linked with the words that are contained in the page, such as the presence of pictures, tables, banners and so on. As mentioned above, the vectorization is based on one context, or one profile, chosen by the user. One may then assume that, for each of the contextes/profiles that have been implemented in the search engine, a list of words that are relevant to that context/profile is available and a related vectorization of the page is stored. Clearly, many refinements to this simple approach may, and should, be considered. First of all, one should not consider simple "words", but rather sets of words (synonyms, singular/plurals, etc...) that all contribute to the occurrence count stored in one component of the vector. In standard information retrieval this is dealt with via stemming (conflation) and a thesaurus.

The dimension m of the vector (that is, the number of relevant words associated to a context) is not theoretically limited to be particularly small, but one must keep in mind that in order to apply the method described over a significant number of pages is it reasonable to consider $m \leq 100$.

According to results of recent works [6, 7], that have demonstrated that the use of the structures of HTML document improves the effectiveness of retrieving HTML documents, several extensions to the simple occurrence count have also been considered in the project partially described in this paper. Here we mention only that a method that identifies also the portion of the web page where a word appears (Title, Text, Link, etc...) has been designed to produce more accurate vectors associated to a page and to a list of words.

We have used several methods to identify the list of m words that are associated with a context/profile, that can be divided into two main groups:

1 creation of the list from user knowledge;

2 extraction of the list from given sets of pages.

In the first case, the words are determined in a setup phase, when the search engine managers decide which are the contextes/profiles supported and what are the words that are representative of that context/profile. This operation may be accomplished together with the users of the engine itself when a thematic engine is devoted to a specific environment (such as an association of companies, a large corporation, a community of users).

In the second case, the words are identified starting from an initial set of pages, that are used as training sample for a context/profile. From the given set of pages one extracts the words with larger total occurrence in the set, and select from the list significant words (skipping articles, verbs, etc...). In standard information retrieval this is usually dealt with a stop word list. Many on-line vocabulary capable of performing this operation automatically are presently available, e.g., Wordnet (http://www.cogsci.princeton.edu/ wn/) and Roget's (http://www.bartleby.com/62/). This approach is particularly significant in the case of the user profiles: in fact, one may consider as a training sample for a profile the pages that have been visited more recently by the user(s) that belong to that profile. This would allow the words associated with the profile to evolve with the behavior of the users in a smooth way.

The latter approach may be used to determine the relevant keywords for a set of pages on-line: when a query returns a set of results, the m most frequent valid words can be identified and then used to build the vectors. That operation can be done for a relevant number of pages once all valid words appearing in each page have been properly stored in a database when the page is first acquired in the engine.

4. Page clustering

Document clustering in information retrieval is usually dealt with agglomerative hierarchical clustering algorithms or k-means algorithm.

Hierarchical agglomerative clustering algorithms [20] create a hierarchy of clusters, that may be viewed as a tree where each node is a cluster of objects and its children form a complete partition of that cluster. At start, a cluster for each object is created; then the method iterates through the cluster set by selecting the closest pair of clusters and merging them together forming a new cluster that replaces the two old clusters in a single one. The procedure is executed until all objects are contained within a unique cluster; then, clusters are selected by setting a threshold level on the inter-clusters distance. These methods attempt to minimize the sum of squares (SS) of the two clusters that can be formed at each step.

The k-means algorithm [9] starts by defining k cluster centroids or *seeds* and assigns each object to the cluster whose centroid is closest. Then, the algorithm is iterated, recomputing the cluster centroids and assigning the objects until a stable clustering is reached. The final solution depends on the choice of the initial centroids and on the parameter k, that determines the number of final clusters and has to be defined a *priori*.

In this study we have initially considered six different hierarchical agglomerative clustering methods that differ in the way they compute distances between clusters (single linkage, complete linkage, group average, weighted pair-group average, centroid, weighted centroid, and Ward's method) and a standard implementation of k-means. While agglomerative hierarchical clustering algorithms are very slow when applied to large document databases [40] (single link and group average methods take $O(|P|^2)$ time, complete link method take $O(|P|^3)$ time), k-means is much faster (its execution time is $O(k \cdot |P|)$).

For the latter, we have tested 3 different techniques for the determination of initial centroids: the first one selects the initial centroids by randomly choosing k documents from the data set,

in the second the initial centroids are determined by randomly generating the elements of each centroid-vector, while in the third one the first centroid is randomly selected and a new centroid is iteratively selected from the other objects maximizing the total distance from the centroids already selected.

Measuring clustering effectiveness and comparing performance of different algorithms is a complex task, and there are no completely satisfactory methods for comparing the quality of the results of a clustering algorithm. In the experiments presented in this work we have used the Calinski-Harabasz $(C-H)$ $pseudo-F$ statistic as a measure of clustering quality; the higher the index the better the cluster quality. For a given clustering, the mathematical expression of the $pseudo-F$ statistic is:

$$C-H = \frac{R^2}{(k-1)} / \frac{\left(1-R^2\right)}{(n-k)},$$

where:

- $R^2 = (SST - SSE)/SST$;
- SST is the sum of the squared distances of each object from the overall centroid, i.e., $\sum_{i=1}^{|P|} \sum_{j=1}^{k} (x_{ij} - \bar{x})^2$ where x_{ij} is the value of the object i in the cluster j, and $\bar{x}$ is the value of the overall centroid;
- SSE is the sum of the squared distances of each object from to the centroid of its own group, i.e., $\sum_{i=1}^{|P|} \sum_{j=1}^{k} (x_{ij} - \bar{x}_j)^2$, where x_{ij} is defined as in SST and $\bar{x_j}$ is the value of the centroid of cluster j.

As a direct measure of the quality of clustering, this statistic is often used to identify the number of clusters in hierarchical methods. In a simulation study involving 30 different halting criterions for cluster analysis, Milligan and Cooper [29] found that the $pseudo-F$ statistics was the one recovering the correct number of groups most often.

Table 1 summarizes some of the experiments conducted on data sets of real web documents. We have used 45 words for page vectorization; the results presented refer to two data sets containing 500 web pages and one containing 2633 pages (DS500_1 and DS500_2, DS2633 in the table). The number of clusters is fixed to 10. The last three rows of the table refer to the different methods to select the initial seeds for k-means described above. The results indicate that the quality of the k-means clustering with the first type of seed (k-Means Bar.1 in the table), as measured by the $pseudo-F$ statistic, is always comparable with the one of the other slower methods. Considering the nature of our application, we have adopted k-Means Bar.1 due to its lower computational complexity.

5. The role of genetic algorithms

Our aim is to select a small subset P' of the original set of pages P for which the sum of the score is large but also where the similarity amongst the selected page is restrained.

We select such a subset using a genetic algorithm (GA). Several reasons motivate this choice. First, the use of metaheuristic techniques is well established in optimization problems where the objective function and the constraints do not have a simple mathematical formulation. Second, we have to determine a good solution in a small computing time, where the dimension of the

	DS500_1		DS500_2		DS2633	
Method	Pseudo F	Time	Pseudo F	Time	Pseudo F	Time
Single Link	78.15	3	60.24	3	170.31	832
Complete Link	103.72	4	78.70	4	230.26	759
Group Average	80.64	4	62.36	3	208.82	676
Weighted Average	71.23	3	61.72	4	307.15	1448
Unweighted Centroid	62.48	4	62.36	4	165.63	1012
Weighted Centroid	78.72	4	60.24	3	267.12	533
Ward's Method	95.10	4	62.72	4	388.28	1286
k-Means Bar.1	74.05	1	66.58	1	337.48	18
k-Means Bar.2	99.12	1	65.73	1	293.28	20
k-Means Bar.3	71.16	1	71.51	2	200.98	21

Times are expressed in seconds of CPU, PC with 1Mhz processor

Table 1: Comparison of clustering algorithms. The number of clusters is fixed to 10

problem may be significantly large. Third, the structure of our problem is straightforward representable by the data structure commonly used by a GA.

GA (see e.g. [4, 14, 18, 28]) are local search algorithms that start from an initial collection of strings (a population) representing possible solutions of the problem. Each string of the population is called a chromosome, and has an associated value called the *fitness function* (ff) that contributes in the generation of new populations by means of genetic operators. Every position in a chromosome is called a *gene* and its value is called the *allelic value*. This value may vary on an assigned allelic alphabet; often the allelic alphabet is {0,1}. At each generation, the algorithm uses the fitness function values to evaluate the survival capacity of each string i of the population using simple operators in order to create a new set of artificial creatures (a new population) which try to improve on the current ff values by using pieces of the old ones. We summarize below the operators adopted.

- **Reproduction.** Individual strings are copied according to their objective function (ff) values. This represents a measure of the utility or goodness related to what we want to maximize. Copying strings according to their fitness function values means that strings with a high value have a higher probability of contribution to one or more offspring in the next generation.

- **Crossover.** The second operator denoted (simple) crossover works as follows: the members reproduced in the new mating pool are mated randomly and afterward each pair of strings, say a and b undergoes a cross change. In order to do this, an integer position k (cut point) is selected uniformly at random along the string. Splitting the two selected strings at this cut point produces a left part and a right part. An offspring can then be produced for example by joining the left part of a's encoding to the left part of b's encoding, or the right part of a's encoding to the right part of b's.

- **Mutation.** The mutation operator plays a secondary role with respect to reproduction and crossover operators. Nevertheless mutation is needed to prevent an irrecoverable loss of potentially useful information which occasionally reproduction and crossover can cause. This operator is an occasional random alteration, of the allelic value of a chromosome that occurs with small probability.

10.

Here follows the outline of the basic genetic algorithm:

1. Generate random population of n chromosomes (suitable solutions for the problem).
2. Evaluate the fitness ff of each chromosome x in the population.
3. Create a new population by repeating the following steps until the new population is complete:
   - Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
   - With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
   - With a mutation probability mutate new offspring at each locus (position in chromosome).
   - Place new offspring in the new population.
4. Use new generated population for a further run of the algorithm.
5. If the end condition is satisfied, stop, and return the best solution in current population.
6. Go to step 2.

In the following we discuss the algorithm engineering.

- **The population**. Starting from the clusters obtained, we define the chromosomes of the initial population as subsets of pages of bounded cardinality (in the GA terminology, a page is a *gene*).

  The genetic algorithm works on the initial population ending up with a representative subset of pages to present to the user. The idea is to start the genetic evolution with a population that is already smaller than the initial set of pages P. Each chromosome is created by picking a page from each cluster, starting with the ones with higher score. Thus, the first chromosome created will contain the pages with the highest score in each cluster, the second chromosome will contain the second best, and so on. If the cardinality of a cluster is smaller than the number of chromosomes to be created, then that cluster will not be represented in each chromosome, while other clusters with higher cardinality may have more that one page representing them in some chromosome.

  We indicate with dc the number of pages included in each chromosome in the initial population, and with nc the number of chromosomes. The population will thus contain $np = dc \cdot nc$ pages.

  Independently from what is the cluster to which the chromosome belongs, the allelic alphabet of its genes is formed by two binary elements, i.e., 0 and 1. Each gene represents a page and the values 0 or 1 assumed by the gene define whether that page is in the pool of pages represented by that chromosome or not. In other words we can say that a chromosome $l = 1, \ldots, ng$ is a vector whose entries are 0 or 1 according to the presence or the absence of that page in that chromosome.

  Here follows a simple example. We have a set of eight pages, i.e., $P = \{1, \ldots, 8\}$, and the page clustering process with $k = 2$ (i.e., with two clusters) gives the following result: pages 1, 2, 3 and 4 are in the first cluster, while pages 5, 6, 7 and 8 are in the second cluster. Assuming, without loss of generality, that $score(i) \geq score(j)$ for all $i, j \in P$, an initial population of four chromosomes can be constructed as depicted in Figure 2.

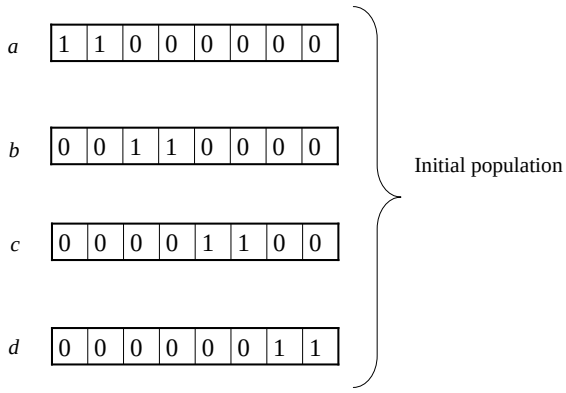

Figure 2: An example of initial population formed by four chromosomes from a set of eight pages

- **The fitness function.** The fitness function computed for each chromosome is expressed as a positive value that is higher for "better" chromosome, and is thus to be maximized. It is composed of three terms.

  The first is the sum of the score of the pages in chromosome C, i.e.,

  $$t_1(C) = \sum_{p_i \in C} score(p_i),$$

  where $score(p_i)$ is the original score that is given to page p_i as described at the beginning of Section 2. This term considers the positive effect of having as many pages with high score as possible in a chromosome, but would also reward chromosomes with many pages regardless of their score (a chromosome with many pages with low score could produce a higher fitness of another with few good pages).

  This drawback is balanced with the second term of the fitness function, that penalizes the distance of the dimension of a chromosome from an *ideal* dimension. Let ID be such ideal dimension; the ratio

  $$t_2(C) = \frac{np}{abs(|C| - ID) + 1}$$

  constitutes the second term of the fitness function: it reaches its maximum np when the dimension of C is exactly equal to the ideal dimension ID, and rapidly decreases when the number of pages contained in chromosome C is smaller or greater than ID.

  The chromosome that are present in the initial population are characterized by the highest possible variability as far as the clusters to which the pages belong are concerned. Although, the evolution of the population may alter this characteristic, creating chromosomes with high fitness where the pages belong to the same cluster and are very similar to each other. Moreover, the fact that pages belonging to different clusters are different in the vectorized space may not be guaranteed, as it depends both on the nature of the data and on the quality of the initial clustering process. For this reason, we introduce in the fitness function a third term, that measures directly the overall dissimilarity of the pages

in the chromosome. Let $D(p_i, p_j)$ be the Euclidean distance of the vectors representing pages p_i and p_j. Then,

$$t_3(C) = \sum_{p_i, p_j \in C, p_i \neq p_j} D(p_i, p_j)$$

is the sum of the distances between the pairs of pages in chromosome C and measures the total variability expressed by C.

The final form of the fitness function for chromosome C is then

$$ff(C) = \alpha \cdot t_1(C) + \beta \cdot t_2(C) + \gamma \cdot t_3(C)$$

where α, β and γ are parameters that depend on the magnitude of the initial score and of the vectors that represent the pages. In particular, α, β and γ are chosen so as the contributions given by $t_1(C)$, $t_2(C)$ and $t_3(C)$ are balanced. Additionally, they may be tuned to express the relevance attributed to the different aspects represented by the three terms.

The goal of the GA is to find, by means of the genetic operators, a chromosome C^* such that $ff(C^*) = \max_{C=1,\ldots,nc} ff(C)$.

6. Experimental results

We have run a number of experiments with the method described above. The code implementing the clustering and the GA was developed in the standard C language under a Windows NT platform. The complete solution process for the instances described below took at most few seconds using a commercial PC with 1 GHz processor and 128 MHz RAM. The experiments are presented below in two subsections, the first devoted to experiments with randomly generated data, and the second dealing with a real application of the proposed method in an experimental thematic search engine.

6.1. Tests on randomly generated data

To verify the behavior of the method, We have generated at random a set of instances of different dimensions in the following way. Given the dimension of set P, say np, and the number of keywords (m) used for vectorization, we determine the matrix of occurrences with np rows and m columns in two steps. First, we fix a value ranging from 0.1 to 0.2 that expresses the density of the matrix (md). Then, we generate the entries of the matrix at random in the interval $(1 - 50)$ according to the uniform distribution, respecting the constraint on the density. The score of each page is itself determined at random and between 0 and 100. The main parameters of the algorithm have been tuned on a few test cases and then were fixed for all the experiments presented, in order to focus the variability of the results on the performance of the method for a given set of parameters rather than on the effect of varying the parameters. The ideal size of the subset of pages selected was set to 10 as to represent a typical output page of a standard search engine. As far as the values of the parameters are concerned, we have that:

- the number of clusters was set to 5 as well as the dimension of the initial chromosome;

- the total number of chromosomes was set to 40, resulting in total of $200 = 5 \cdot 40$ pages composing the initial population;
- α was set to 1;
- β was set to 50;
- γ was set to 1.

The first set of experiments was designed to test the behavior of the GA in determining a reduced set of pages with high fitness. The genetic algorithm that we have designed composes new chromosomes with high fitness in few iterations. Then, the highest fitness is seldom increased in the following iterations, despite the crossover and the mutation operations. This is shown in Table 2, where we report the increase in the fitness function value determined by the algorithm and the iterations at which the best chromosome has been created.

The total number of iterations was bounded to 1000. We have considered instances with 500, 1000, and 10000 pages. In Table 2 the first column contains the name of the experiment, that reports the size of the initial set of pages and an index associated with each random replication of a data set with same dimensions.

From the results of Table 2 we see that a significant increase in the value of the fitness function is always determined (column six), and, moreover, the number of iterations needed to identify the best solution is small and often very small. Thus, we have reasonably resorted to halting the algorithm once the best fitness value has remained the same for the last 50 iterations.

In the second set of experiments, we have focused on the evaluation of the results obtained. For each experiment, we have compared the subset of pages determined by our method (referred to, in the following, as *genetic subset*) with the subset of the same dimension containing the best pages according to the initial score (referred to, in the following, as *standard subset*). Assume that the method has determined a subset of h pages to present to the user. We question what we lose and what we gain in presenting this set of pages instead of the best h pages according to the score, as in a standard query result. To assess the difference between the two sets, we consider and compare the following measures:

a) the sum of the score of the pages in the subset;

b) number of clusters represented in the subset (a cluster is represented in a subset if at least one page belonging to that cluster is in the subset);

c) the average Euclidean distance amongst all pairs of pages in the subset;

d) the average Hamming distance amongst all pairs of pages in the subset (Hamming distance between two pages is computed as the sum of keywords for which some occurrences are present in one page and not present in the other).

While a) measures the quality of the subset according to page score, term b) aims to evaluate how the final subset takes into account the structure of the initial set of pages (represented by the clusters). Terms c) and d) are both indirect measures of the dissimilarity among the pages in the subset. While the average Euclidean distance is itself weighted in the fitness function used in the GA, we have introduced the additional measure of dissimilarity based on the Hamming distance to make the evaluation of the quality of the subset presented more stable. In fact,

Experiment	number of keywords	matrix density	best fitness at iteration 0	best fitness overall	fitness increase ratio	best iteration
EXP500_1	20	0.10	1540.10	6115.79	2.97	368
EXP500_2	20	0.15	1549.64	6122.12	2.95	217
EXP500_3	20	0.20	1564.93	6140.98	2.92	133
EXP500_4	30	0.10	1564.81	6130.30	2.92	368
EXP500_5	30	0.50	1560.14	6137.34	2.93	48
EXP500_6	30	0.20	1568.80	6143.90	2.92	48
EXP500_7	50	0.10	1574.39	6141.66	2.90	48
EXP500_8	50	0.15	1567.95	6132.94	2.91	528
EXP500_9	50	0.20	1587.64	6171.67	2.89	49
EXP1000_1	20	0.10	1547.19	6127.88	2.96	205
EXP1000_2	20	0.15	1549.15	6129.54	2.96	165
EXP1000_3	20	0.20	1561.25	6148.16	2.94	46
EXP1000_4	30	0.10	1564.10	6148.63	2.93	56
EXP1000_5	30	0.50	1567.21	6154.22	2.93	194
EXP1000_6	30	0.20	1572.41	6164.46	2.92	51
EXP1000_7	50	0.10	1574.04	6146.77	2.90	99
EXP1000_8	50	0.15	1581.33	6172.41	2.90	48
EXP1000_9	50	0.20	1595.10	6191.06	2.88	79
EXP10000_1	20	0.10	1550.91	6149.68	2.86	988
EXP10000_2	20	0.15	1550.91	6149.68	2.96	50
EXP10000_3	20	0.20	1550.91	6149.68	2.96	50
EXP10000_4	30	0.10	1571.79	6170.93	2.93	88
EXP10000_5	30	0.50	1575.34	6185.65	2.93	349
EXP10000_6	30	0.20	1581.10	6173.52	2.90	545
EXP10000_7	50	0.10	1581.06	6182.25	2.91	198
EXP10000_8	50	0.15	1603.56	6205.47	2.87	227
EXP10000_9	50	0.20	1610.41	6218.97	2.86	988

Table 2: Behavior of the genetic algorithm for different randomly generated data sets

although the two measures are likely to be correlated, the second one has been often used as a consistent indicator of similarity between documents.

In the results of Table 3 we have fixed the number of keywords at 30 and the density of the matrix of occurrences to 0.15, and have instead replicated different randomly generated instances for 500, 1000, and 10000 pages. The last three columns of the table report the percentage of the variation obtained with the *genetic subset* with respect to the *standard subset*. A positive value means that genetic is better than standard, while a negative one indicates the opposite.

It is evident from the results of Table 3 how the solutions identified by our method provide a significant gain in page dissimilarity (measured both by Euclidean and Hamming average distance, columns five and six of the table) while the loss in total score with respect to the simple ordering is negligible (column four in the table).

A third set of experiments (see Table 4) has been designed to evaluate the performance of the algorithm when duplicated pages are present. For the same setting of the parameters as above, for different dimensions of the initial set of pages, we have considered the same instance and

have artificially duplicated the page with best score once, twice, and 3 times. The objective of the experiments is to assess the ability of the method of excluding duplications from the final results. The figures associated to the measures of dissimilarity between pages in Table 3 are promising: in fact, they show that, when duplications are present, the total variability of the subsets selected by our algorithm increases with respect to the variability of the subset determined with the simple ordering. This effect, being the data set exactly the same apart from the duplicated pages, is to be attributed to the fact the the duplications appear in the classic ordering while are excluded by the GA in our solutions. This behavior is more evident when the number of duplicated pages is increased.

6.2. Tests on real data

The method described in this paper has been developed in the framework of a project funded by the Italian Ministry of Industry; within the same project a complete thematic search engine has been realized and is now being integrated with the new search method.

Using this tool, a thematic database of web pages on "Infancy & Childhood" was collected through spidering Italian web sites in the first months of 2003. Out of this database we have extracted $3,710$ pages, that were the result of a specific search on the database of the engine.

The pages were then vectorized according to 57 *macro keywords*, where each such keyword was effectively the collection of the occurencies of root words and strong synonyms; this determined a *vector* of 57 integers for each page that described that page for the scopes of our application.

The $3,710$ pages were then sorted according to five different types of user score, corresponding to different preference criteria expressed by test users in terms of the macro keywords. The algorithm was set up selecting a total of 200 choromosomes each with 5 genes, resulting in an initial "pool" of the best $1,000$ web pages according to the specific user score adopted. Then, the algorithm was run to find the best chromosome, i.e., the best subset of pages, according to the procedures described in the previous sections.

The results of the experiments are presented in Table 5. The columns of the table have the same format of the tables in Section 6, reporting the dimension of the best chromosome, the number of clusters represented, and the variations of i) total user score ii) average Hamming distance and iii) average Euclidean distance, between the pages in the selected chromosome and the pages obtained by the simple ordering of the user score.

Also in this case the behavior of the algorithm is interesting: small reductions in the total score of the subset are balanced by high increases in the clusters representativeness and internal dissimilarity in the selected pages.

Experiment	dimension of best chromosome	cluster represented standard	genetic	total score %var.	avg. Hamming distance % var.	avg. Euclidean distance % var.
EXP500_1	11	3	5	-1.7	22.5	1.3
EXP500_2	11	5	5	-1.5	21.6	0.3
EXP500_3	11	3	5	-5.4	13.1	12.7
EXP500_4	11	4	5	-1.7	4.7	0.0
EXP500_5	11	4	5	-3.1	20.3	2.7
EXP500_6	11	5	5	-1.5	21.6	27.0
EXP500_7	11	5	5	-2.2	6.1	5.4
EXP500_8	11	5	5	-2.4	10.3	1.3
EXP500_9	11	5	5	-2.1	35.7	17.9
EXP500_10	11	2	4	-3.9	21.6	8.4
average_500	11	4.1	4.9	-2.5	17.7	7.7
EXP1000_1	11	3	5	-1.5	23.7	16.1
EXP1000_2	11	4	5	-0.2	1.6	5.4
EXP1000_3	11	3	5	-0.8	5.3	5.7
EXP1000_4	11	4	5	-0.6	6.8	15.0
EXP1000_5	11	5	5	-1.6	6.8	17.8
EXP1000_6	11	3	5	-1.5	0.0	14.8
EXP1000_7	11	4	5	-2.2	14.5	24.7
EXP1000_8	11	5	5	-0.9	28.0	22.5
EXP1000_9	11	5	5	-0.8	20.2	8.3
EXP1000_10	11	4	5	-0.6	11.4	16.9
average_1000	11	4	5	-1.1	11.8	14.7
EXP10000_1	11	4	5	-0.1	16.5	16.9
EXP10000_2	11	4	5	0.5	14.7	15.7
EXP10000_3	11	4	5	0.0	5.7	7.2
EXP10000_4	11	3	4	-0.6	0.2	19.0
EXP10000_5	11	4	5	-0.1	31.7	16.9
EXP10000_6	11	5	5	-0.1	23.7	24.3
EXP10000_7	11	4	5	-0.5	44.1	34.6
EXP10000_8	11	3	5	-0.4	36.3	21.1
EXP10000_9	11	4	3	-0.3	69.2	38.0
EXP10000_10	11	4	5	-0.4	32.3	42.0
average_10000	11	3.9	4.7	-0.2	27.4	23.6

Table 3: Evaluation of results on randomly generated instances

Experiment	dimension of best chromosome	number of duplicates	total score % var.	avg. Hamming distance % var.	avg. Euclidean distance % var.
EXP500_1	11	0	-2.2	6.1	5.4
EXP500_2	11	1	-1.9	21.4	8.0
EXP500_3	11	2	-2.1	28.3	21.3
EXP500_4	11	3	-1.8	16.8	8.3
EXP1000_1	11	0	-1.0	7.9	4.3
EXP1000_2	11	1	-0.9	10.25	9.7
EXP1000_3	11	2	-0.8	12.3	8.1
EXP1000_4	11	3	-1.1	21.8	21.7
EXP10000_1	11	0	-0.4	30.2	33.3
EXP10000_2	11	1	-0.4	34.2	34.2
EXP10000_3	11	2	-0.4	58.9	52.8
EXP10000_4	11	3	-0.4	58.9	56.6

Table 4: Evaluation of results on randomly generated instances with page replications

Experiment	dimension of best chromosome	cluster represented		total score %var.	avg. Hamming distance % var.	avg. Euclidean distance % var.
		standard	genetic			
user score 1	10	1	5	-7.77	35.38	747.61
user score 2	11	2	5	-9.09	16.97	433.92
user score 3	10	1	5	0.00	23.58	980.13
user score 4	11	3	5	-10.82	0.10	266.01
user score 5	10	1	5	-5.83	32.19	646.29

Table 5: Results obtained on a database of 3710 web pages

7. Conclusions

The experimental results presented in this paper show that the proposed method can be effective in the selection of small subsets of pages of good quality, where quality is not considered as a simple sum of the quality of each page but as a global characteristic of the subset. The implementation of the GA and of the clustering algorithm have allowed us to obtain convergence to solutions in reasonably short computational times on a standard personal computer (a few seconds).

One may question whether the described method can be run *on-line* in a search engine as the standard execution of a user's query. We believe that with proper bounds on the parameters and a proper engineering of the algorithms deployed, the computational effort can be dealt with and that only few seconds may be added to the overall search process. Future work will cover the extension of the page vectorization technique and the definition and test of automatic procedures for parameter tuning in the genetic algorithm.

References

[1] Aggarwal, C.C., Gates, S.C., Yu, P.S.: On the merits of building categorization system by supervised clustering. In: Proc. of the 5th ACM SIGKDD International Conference on Information and Knowledge Management, Bethesda, Maryland,USA, November 1998, 3–7.

[2] Boley, D., Gini, M., Han E., Hastings K., Karypis G., Kumar V., mobasher B., Moore J.: Partitioning-based clustering for Web document categorization. Decision Support System, V. 27 (1999), 329–341.

[3] Boughanem, M., Chrisment, C., Tamine, L.: Genetic Approach to Query Space Exploration. Information Retrieval, 1, 1999, 175–192.

[4] Chambers, L.: Practical Handbook of Genetic Algorithms: Applications. CRC Press, Boca Ratoon, Florida (1995).

[5] Chen, H.: Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms. Journal of the American Society for Information Science, V.46 N.3, 1995, 194–216.

[6] Cutler, M., Deng, H., Maniccam, S.S., Meng, W.: A New Study on Using HTML Structures to Improve Retrieval. In: Proc of the11th IEEE International Conference on Tools with Artificial Intelligence.

[7] Cutler, M., Shih, Y., Meng, W.: Using the Structure of HTML Documents to Improve Retrieval. USENIX Symposium on Internet Technologies and Systems (NSITS'97), Monterey, California, 1997, pp. 241–251.

[8] Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W.: Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In: Proc. of the 15th Annual International ACM/SIGIR Conference, Copenhagen, 1992.

[9] Dubes, R.C., Jain, A. K.: Algorithms for Clustering Data. Prentice Hall, 1988.

[10] Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, New York (1973).

[11] Felici, G., Sun, F.S., Truemper, K.: A Method for controlling errors in two-class classifications. In: COMP99: 23rd Annual International Computer Software and Applications Conference, Phoenix, Arizona, IEEE Computer Society, Los Alamitos, CA, (1999) 186–191.

[12] Glover, E.J., Lawrence, S., Gordon, M.D., Birmingham, W.P., Giles, C.L.: Recommending Web Documents Based on User Preferences. SIGIR 99 Workshop on Recommender Systems.

[13] Glover, E.J., Lawrence, S., Gordon, M.D., Birmingham, W.P., Giles, C.L.: Web Search – Your Way. Communications of the ACM.

[14] Goldberg, E.D.: Genetic Algorithms in Search Optimization & Machine Learning. Addison-Wesley Publishing Company (1999).

[15] Gordon, M.: Probabilistic and Genetic Algorithms for Document Retrieval. Communications of the ACM, V.31 N.10, 1208–1218.

[16] Gordon, M.: User-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm. Journal of the American Society for Information Science,V.42 N.5, 31–322.

[17] Hearst, M.A., Pederson, J.O.: Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In: Proc. of ACM SIGIR '96, August, 1996, Zurich.

[18] Holland, J. H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI (1975).

[19] Horng, J.-T., Yeh C.-C.: Applying genetic algorithms to query optimization in document retrieval. Information Processing and Management 36 (2000), 737–759.

[20] Jain, A.K, Murty,M.N., Flynn, P.J.: Data Clustering: a Review. ACM Computing Surveys, Vol.31, n.3, 1999, pp.264-323.

[21] Kohonen, T.: Self-Organizing and Associated Memory. Springer, Berlin (1988).

[22] Kohonen, T.: Exploration of very large database by self-organizing maps.In: Proc. of the IEEE International Conference on Neural Networks, V. 1, (1997) 1–6.

[23] Koller, D., Sahami, M.: Hierarchically classifying documents using very few words. In: Proc. of the 14th International Conference on Machine Learning, Nashville, Tennessee, July 1997, 170–178.

[24] Kraft, D.H., Petry, F.E., Buckles, B.P., Sadavisan, T.: Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback. In Sanchez E., Zadeh L.A., Shibata T. (Eds.), Genetic Algorithms and Fuzzy Logic Systems, Soft Compuiting Perspectives. World Scentific,1997, 155–173.

[25] Leuski, A.: Evaluating Document Clustering for Interactive Information Retrieval. In: Proc. of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 2001, 33–44.

[26] Lin, K., Kondadadi, R.: A Similarity-Based Soft Clustering Algorithm for Documents. In: Proc. of the Seventh International Conference on Database Systems for Advanced Applications, April, 2001.

[27] Lin, K., Kondadadi, R.: A Word Based Soft Clustering Algorithm for Documents. Proceedings of the 16th International Conference on Computers and Their Applications, March, 2001

[28] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer Verlag, Berlin Heidelberg (1992).

[29] Milligan, G.W., Cooper, M.: An examination of procedures for determining the number of clusters in a data set. Psychometrika, V.50 N.2, 159–179.

[30] Petry, F., Buckles, B., Prabhu, D., Kraft, D.: Fuzzy Information Retrieval Using Genetic Algorithms and Relevanc Feedback. In: Proceedings of the ASIS Annual Meeting, 1993, 122–125.

[31] Sanchez, E., Pierre, Ph.: Fuzzy Logic and Genetic Algorithms in Information Retrieval.In: Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Net and Soft Computing, Iizuka, Japan, 1994, 29–35.

[32] Steinbach, M., Karypis, G., Kumar, V.: A comparison of Document Clustering Techniques. In: Proc. of Text Mining Workshop, KDD, (2000).

[33] Van Rijsbergen, C.J.: Information Retrieval. Butterwhorts, London, 1979. Second edition.

[34] Voorhees, E.M.: The cluster hyphotesis revisited. In: Proc. of ACM SIGIR, June 1985, 188–196.

[35] Willet, P.: Recent trends in hierarchical document clustering: A critical review. Information Processing & Management, V.24 N.5 (1988), 577–597.

[36] Yang, J.J., Korfhage R.R., Rasmussen, E.: Query Improvement in Information Retrieval Using Genetic Algorithms: a Report on the Experiments of the TREC Project. In: Proceedings of the 1st Text REtrieval Conference (TREC-1), 1993, 31–58.

[37] Yang, J.J., Korfhage R.R.: Query Optimization in Information Retrieval Using Genetic Algorithms. In: Proceedings of the 5th ICGA, 1993, 603–613.

[38] Zacharis, Z. N., Panayiotopoulos, T.: Web Search Using a Genetic Algorithm. IEEE Internet Computing, V.5 N.2:18–26, March/April 2001.

[39] Zamir, O., Etzioni, O. : Grouper: a dynamic clustering interface to web search results. WWW8/Computer Networks, V.31 N. 11-16, (1999) 1361–1374.

[40] Zamir, O., Etzioni, O.: Web Document Clustering: A Feasibility Demonstration. In: Proc. of 19^{th} international ACM SIGIR conference on research and development in information retrieval (SIGIR 98), 46–54.

[41] Zamir, O., Etzioni, O., Madani, O., Karp, R.M.: Fast and Intuitive Clustering of Web Documents. KDD '97, (1997), 287–290.