T. AMGHAR
B. LEVRAT
F. SAUBION
Université d'Angers
Leria, France
{amghar,levrat,saubion}@info.univ-angers.fr}

Using genetic algorithms to compose documents: A way to focus attention on relevant information

Abstract

One critical step in information retrieval is the skimming of the returned documents, considered as globally relevant by an Information retrieval system as responses to a user's query. This skimming has generally to be done in order to find the parts of the returned documents which contain the information satisfying the user's information need. This task may be particularly heavy when only small parts of the returned documents are related to the asked topic. Therefore, our proposition here is to substitute an automatic extraction and recomposition process in order to provide the user with synthetic documents, called here composite documents, made of parts of documents extracted from the set of documents returned as responses to a query. The composite documents are built in such a way that they summarize as concisely as possible the various aspects of relevant information for the query and which are initially scattered among the returned documents.

The combinatorial aspect of the problem and its modeling characteristics led us to consider resolution paradigms that are commonly used in the combinatorial optimization community. Moreover, viewing the generation of composite documents as the result of a succession of segmentations and recombinations of documents until a satisfaction criterion threshold is reached makes this task suitable for a treatment by evolutionary algorithms, and more precisely by genetic algorithms: individuals are texts and the fitness function aims at optimizing a satisfaction criterion based on similarity. We have implemented several variants of the algorithm and we proposed an analysis of the first experimental results which seems promising for a preliminary work.

1. Introduction

One critical step in information retrieval is the skimming of the returned documents, considered as globally relevant by an Information retrieval system [1,10,6] as responses to a user's query. This skimming has generally to be done in the aim to find the parts of the returned documents which contain the information satisfying the user's information need. This task may be particularly heavy in the case where only small parts of the returned documents are related to the asked topic. For example, it may be the case when a general document base or a set of press agency dispatches is asked for technical questions: some documents may incidentally contain relevant information but, as they have not been written to satisfy these kinds of topics, only small parts of them can eventually contain the suitable information.

Therefore, our proposition here is to furnish a response to a query not directly in terms of documents of the base taken in their wholes, but rather by producing new documents, extracted from the initially furnished documents as responses. The built documents, called composite documents, are made of parts of the previous ones, called here segments, and are intended to synthesize or summarize the information relevant for the query. For that purpose, we use an automatic extraction and recomposition process which provide the user with this synthetic documents made of parts of documents relevant for his information need initially scattered among the returned documents. So, the user may focus his attention only on interesting information avoiding the boring task of exploring non relevant parts in the returned documents.

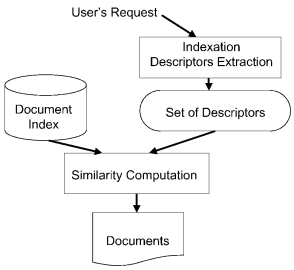

Fig.1. Information retrieval process

The combinatorial aspect of the problem and its modeling characteristics led us to consider resolution paradigms that are commonly used in the combinatorial optimization community. Moreover, viewing the generation of composite documents as the result of a succession of segmentations and recombinations of documents until a satisfaction criterion threshold is reached make this task suitable for a treatment by evolutionary algorithms, and more precisely by genetic algorithms to solve combinatorial problems, a genetic algorithm manages a pool (population) of configurations (individuals) of the problem and its purpose is to generate the best possible configuration, a solution if this notion is appropriate. In our context, the characterization of what constitutes a

solution is a rather difficult task, since it is not only strongly related to the satisfaction degree of the end user, but depends also on what the initial materials are (i.e. the documents base from which it is built). What is searched for is not appropriately said a solution but rather what is the best individual we can built from initial data. So stated, our problem is merely an optimization rather than a solution finding one. In this sense our objective will be to provide good answers with regards to a given multipurpose quality function, taking into account different evaluation points of view. In this genetic algorithm approach, individuals consists of parts of documents and are evaluated according to a fitness function that takes into account its semantics appropriateness with regards to the initial query of the user. The population of documents evolves thanks to recombinations and mutations that correspond to exchanges of text parts between two individuals and the random changes of some these parts. These basic genetic operators produce new individuals that are added to the current population. In order to converge toward a good individual, a selection stage in necessary and consists in eliminating the less performing individuals with regards to the fitness function from the current population. This process is repeated until a maximum number of iterations has been reached. Experiments have been performed on a specific corpus and provide very promising results.

2. Background

2.1 Information retrieval

The vectorial model [9]} constitutes an alternative to the binary model which responds to their major defaults (lack of partial results, lack of a similarity evaluation criterion and lack of ranking of the returned documents). It has been shown as sufficiently simple and efficient compared with other more elaborated models like the extended boolean one [8] or the generalized vector one [11] which are computationally more costly. This is enough to justify this choice as a model for our information retrieval system, since our works mainly bears on the set of the returned documents to a user's query by an information retrieval system, and it doesn't matter what precisely is its underlying model. Let us sketch however here some of the underlying principles of the this model sufficient to the comprehension of our work. The main characteristic we use of this model is that it rends it possible to represent the contribution of the terms to the general semantics of a text by means of non-binary weights. Thus queries and documents representations are expressed in terms of vectors of weights. Each vector component expresses the semantic weight of the term associated to the component and renders both the representativity of the term for the question or the document and its ability to distinguish documents the ones from the others. The application of the vectorial algebra makes it possible to model the semantic similarity between documents or between documents and queries as a vectorial distance expressed, for example, by the angles between the concerned vectors. In this case, this measurement of similarity can be chosen as the cosine between the two vectors in the space of terms considered as having *N* for its dimension if there are *N* terms in the documents base. This cosine similarity criterion has a real value between 0 and 1, collinearity corresponds to the value 1 and indicates the highest similarity degree interpreted as the highest semantic similarity between the compared entities (a query and a document or a document and another document). The weights of the terms *t* pertaining to a document *o* are defined in the following way:

Definition 1. *Weight of term t in an object o*

$$w_0(t) = f_0(t)(1 - \log(\frac{N}{f(t)}))$$

where $f_0(t)$ *is the normalized frequency of the term t in the object o, f(t) is the number of documents containing the term t, N the total number of documents of the corpus,* $f_0(t) = occur_0(t) / occur_0(t_{\max})$ *where* $t_{\max}$ *is the term with the highest occurrence number in the object o, and* $occur_0(t)$ *is the number of times that the term t appears in the object o.*

The similarity between a request *q* and a document *d* or between two documents is also defined as we have previously said as a cosine measurement of the angle between the two associated vectors. More formally, let $\vec{o} \equiv (o_1, ..., o_n)$ the vector corresponding to the object *o* (document or request), with o_i indicating the weight associated with the ith component of *o* (this component is associated with the ith term), then the similarity between the two objects (in fact here, a document *d* and a request *q*) in this model is calculated as follows:

Definition 2. *Similarity*

$$sim(d,q) = \frac{\vec{d}.\vec{q}}{|\vec{d}|.|\vec{q}|}$$

2.2 Segmentation of Documents

The segmentation consists in cutting out documents provided by the document retrieval system in order to extract paragraphs containing elementary pieces of information since, most of the time, a complete document may

include different topics. Our purpose is then to recombine these paragraphs into a more relevant composite document. Segmentation can be considered from several points of view: we may either exploit the layout of the text (morphosyntaxic criteria like punctuation) or detect semantic variations with methods like Text Tiling [3] or Decision Tree. As mentioned in the introduction, we want to reduce the computational effort as much as possible and we have chosen to use a morphosyntaxic segmentation of texts into paragraphs. In the experimental section, we will see that this type of segmentation provides promising results and can be achieved very quickly.

2.3 Genetic Algorithms

Genetic algorithms (GA) [4,5] are mainly based on the notion of adaptation of a population (i.e., a set of individuals) to a criterion using evolution operators like crossover [2]. If individuals are considered as potential solutions to a given problem, applying a genetic algorithm consists in generating better and better individuals with respect to the problem by selecting, crossing, and mutating them. This approach reveals very useful for problems with huge search spaces. We give a general framework for GAs and we refer the reader to [7] for a survey. A GA consists of the following components:

- a representation of the individuals,
- an evaluation function *eval*: the evaluation function rates each potential solution with respect to the given problem,
- genetic operators that define the composition of the children: two different operators will be considered: crossover allows to generate new individuals (the offsprings) by crossing individuals of the current population (the parents), mutation arbitrarily alters one or more genes of a selected individual,
- parameters: population size p_{size} and probabilities of crossover p_c and mutation p_m.

We now precise how this general algorithmic scheme will be instantiated to fit our problem.

3. Designing a Genetic Algorithm for Composite Documents Generation

As mentioned in the introduction, our purpose is to produce composite documents in order to answer to a given query. The general process consists in using first a document retrieval system in order to select a sets of relevant initial documents, to cut them into basic paragraphs and then to combined them, thanks to a specific GA, to obtain a new composite document, that is expected to contain all pertinent informations with regards to the initial query. This full rocess can be thus schemed as in figure 2. We only depict here the GA part that we have designed for this problem.

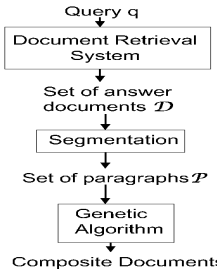

Fig. 2. General process

Representation A crucial point for the design of an efficient evolutionary solving procedure relies on the representation of the search space. In our context, the initial ordered set of documents D generated by the retrieval system, will be cut into a set of basic paragraphs P from elements of D. Therefore, our search space is 2^P. Individuals will be documents of a given size (here, composed of 50 paragraphs) and their set is denoted *I*. A population *p* is a subset of *I*.

Evaluation In the composition of documents, we have to consider several criteria of evaluation. Of course, in order to assess the proximity between a text and the user's query we use a measure of similarity. Here we do not want to evaluate paragraphs independently and we only allow our algorithm to use global similarity of a document. We may also use a specific indicator to rate an individual according to the order of the documents provided by the document retrieval process (i.e., the order of initial documents in D. We first define a function that computes the average rank of an individual *i* with regards to the ranks in D of the documents its paragraphs come from: $rank\ (i) = \frac{1}{n}\sum_{k=1}^{n} r_k$ where r_k the rank in D of the document that originally contains the k^{th} paragraph of *i*. Using the definition of similarity given in definition 2, we define an evaluation function with regards to a given query q. This function takes into account both similarity and rank $Eval\ 1_q(i) = -\log(\frac{1}{rank\ (i)})(sim\ (i,q) - rank\ (i))$ We will also introduce a simple similarity evaluation $Eval\ \ 2_q(i) = sim\ (i,q)$ in order to compare different versions of the algorithm.

Population Management The initial population is randomly generated from the set of all paragraphs in order to generate a set of 100 individuals. At each step, crossover and mutation are applied to generate new individuals that are addedto the current population. The 100 best individuals of the current population are then selected in order to generate the next population and to keep a constant size. This process is repeated until a maximum number of iteration has been reached (this number is fixed here to 100).

Genetic Operators Crossover operators [2] are used in the evolution process to combine individuals from the population and produce new ones. Here we focus on a basic crossover operator, the single point crossover, that takes as input two individuals, the parents, and produces two children by exchanging the values of the parents. The parents are classically selected according to their fitness (i.e., the best evaluated individuals have better chance to participate to a crossover operation). More precisely, crossover is performed in the following way:

select two individuals according to their fitness

- generate randomly a number $r \in [0,1]$
- if $r > p_c$ then the crossover is possible;
  - select a random position $p \in \{1,...,n-1\}$
  - from selected individuals $(a_1,...,a_p,a_{p+1},...,a_n),(b_1,...,b_p,b_{p+1},...,b_n)$, two new individuals $(a_1,...,a_p,b_{p+1},...,b_n),(b_1,...,b_p,a_{p+1},...,a_n)$ are added to the current population.

The mutation is defined as:

For each individual i and for each paragraph k in i, generate a random number $r \in [0,1]$ *if* $r > p_m$ *then mutate k (randomly exchange the paragraph with any element from P).*

For the experiments, p_c is set to 0.2 and p_m=0.02 (empirical settings determined by experimentation).

4. Experiments

The previous GA has been implemented and connected with an existing document processing system.

Extraction of Documents In our approach, we first retrieve, from a corpus of texts, documents corresponding to the user's query with a document search system (see figure 2). These documents are ordered according to their global similarity with the initial query. We use here the corpus of documents of TREC (Text REtrieval Conference: http://trec.nist.gov/) which consists of columns of several newspapers written in ASCII code. Our document retrieval system is a PERL program named PIRES (Perl Information REtrieval System: http://www.info.univ-angers.fr/pub/robin/). This program is based on the classical vectorial model and the documentary research is implemented thanks to a measure of similarity.

Genetic Algorithm According to the different evaluation functions provided in the evaluation section, we have implemented three different versions of the GA that correspond to the three possible evaluations:

GA_1: evaluation $Eval1_q$ including *rank*

GA_2: evaluation $Eval2_q$ using only the similarity

GA_3: blind GA with no evaluation function in order to compare our results with a full random document composition process.

Remind that the main parameters are: individual size = 50, population size = 100, probability of crossover = 0.2, probability of mutation = 0.02 and number of generations = 100.

Comparison Criteria Our purpose is to test and compare the three versions of the genetic algorithm. Our experimental process is the following (according to fig. 1): we fix a query and we extract a set $\mathcal{D}$ of documents from our corpus with PIRES. These documents serve as basis for the text tiling process in order to obtain our set of paragraphs $\mathcal{P}$. After the runs of the GA, the paragraphs are evaluated by hand in order to rate from a user point of view their pertinence with regards to the query. Based on this pertinence evaluation we define a first experimental evaluation criterion (Recall) which is the rate of relevant paragraphs appearing in an individual $NbPert_i$ with regards to the total number of relevant paragraphs which were available in the corpus $NbPert_i$. A second criterion is the similarity between a individual and the initial query *sim(i,q)*. The last criterion (Recall Back) evaluates the covering ability of an individual *i* with respect to the initial set of documents $\mathcal{D}$, i.e. we calculate the average rank r_k of the paragraphs *k* in *i* with regards to the initial order in. $\mathcal{D}$

Experimental Results We test two different queries: Query1 *Oil platforms attacks during the gulf war* and Query2 *Brain drain migrations*. We first give the average results for the three versions of the GA described above. We chose to run each algorithm 100 times and to compute the average values of the criteria previously described over the 15 best individuals obtained after a full run of 100 generations.

	Query 1			*Query 2*		
	GA_1	GA_2	GA_3	GA_1	GA_2	GA_3
Recall	*0.0072*	*0.0058*	*0.0047*	*0.9256*	*0.7606*	*0.6086*
Similarity	*0.9978*	*0.9534*	*0.9306*	*0.9999*	*0.9999*	*0.8721*
Recall Back	*0.0220*	*0.0090*	*0.0068*	*0*	*0*	*0*

This results show the improvements for all the three evaluation measures of GA_1 and GA_2 compared with GA_3 It is interesting to see GA_1 appears to be the more efficient for the three evaluation criteria (it has the particular ability to better cover he initial set of documents (recall back criterion)). Results n the query 2 are due to the extreme parsimony of relevant information for t in the returned documents. Best individuals of each GA versions contain the same relevant segment duplicated so as to constitute a document. This explains the high value of the similarity measure and the value zero for the recall back measure since this segment belong to the first relevant document initially returned by PIRES. We have performed several tests with other queries and observed similar results.

5. Conclusion

In this paper we have presented a new approach to generate composite documents hat provide the user a better overview and a faster exploitation of a set of documents he gets from an information retrieval system. Our techniques based n a GA could be more efficiently tuned and extended in several ways. We could introduce more sophisticated evaluation criteria such as the intrinsic consistency of a composite document, with regards to the relationships of its different pieces of text. The genetic framework could be used to naturally handle such a multi-objective optimisation problem.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval ACM Press/Addison-Wesley, 1999*.

[2] D.E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning Reading, MA:Addison-Wesley, 1989.*

[3] M.A. Hearst. *Texttiling: Segmenting text into multi-paragraph subtopic passages. Computational Linguistics, 23(1):33--64, 1997.*

[4] J. Holland. *Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.*

[5] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems. Phd thesis, University of Michigan, 1975.*

[6] M. Lesk. *Practical Digital Libraries: Books, Bytes, and Bucks Morgan Kaufmann, 1997.*

[7] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd, extended ed.) Springer, 1996.*

[8] G. Salton, *E. Fox, and H. Wu. Extended boolean information retrieval. Commun. ACM, 26(11):1022--1036, 1983.*

[9] G. Salton and M. Lesk. *Computer evaluation of indexing and text processing. J. ACM, 15(1):8--36, 1968.*

[10] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images Van Nostrand Reinhold, 1994.*

[11] M. Wong, W. Ziarko, and P. Wong. *Generalized vector space model in information retrieval. In SIGIR pages 18--25, 1985.*