LEARNING QUERIES FOR A FUZZY INFORMATION RETRIEVAL SYSTEM BY MEANS OF GA-P TECHNIQUES

Oscar Cordón
Dept. of Computer Science and A.I.
University of Granada
18071 - Granada (Spain)
e-mail: ocordon@decsai.ugr.es

Félix de Moya
Dept. of Librarianship
University of Granada
18071 - Granada (Spain)
e-mail: felix@goliat.ugr.es

Carmen Zarco
BARATZ - Servicios de
Teledocumentación
Fuencarral, 123, 3º. Madrid (Spain)
e-mail: mcarmen@baratz.es

Summary

In this contribution, a Genetic Algorithm-Programming algorithm is proposed to automatically learn weighted queries —modeling the user's needs— for a Fuzzy Information Retrieval System. Although the fuzzy retrieval model constitute a powerful extension of the boolean one, the problem is that users are not usually able to express their query requirements in the form of a weighted query including the boolean operators AND, OR and NOT. The proposed model will be able to assist the user to obtain such queries by starting from a set of relevant documents and by applying an off-line adaptive process.

Keywords: Fuzzy Information Retrieval, GA-P algorithms, weighted queries, automatic query learning.

1 Introduction

Information Retrieval (IR) may be defined, in general, as the problem of the selection of documentary information from storage in response to search questions provided by an user [11]. Information Retrieval Systems (IRSs) are a kind of Information Systems that deal with data bases composed of information items —documents that may consist of textual, pictorial or vocal information— and process user queries trying to allow the user to access to relevant information in an appropiate time interval.

Most of the commercial IRSs are based on the boolean IR model [11], which presents some limitations. One of them is that boolean IRSs lack the ability to deal with imprecision and subjectivity, both of which are inherently present in the IR process. This is why the Fuzzy Information Retrieval (FIR) paradigm [3] has emerged as an active research area in the past decade. FIRSs present a query language that allows the user to build queries composed of statements of positive or negative terms —numericaly or linguisticaly weighted— joined by boolean operators AND and OR. This query structure allows the system to improve the retrieval activity solving the boolean model problems.

Nevertheless, the use of weights requires a clear knowledge of the query semantics to translate a fuzzy concept into a crisp numeric value in [0,1]. On the other hand, both in boolean and FIRSs, it is difficult for non expert users to express their retrieval needs in the form of a query composed of different statements joined by the logical operators AND and OR. To solve these two problems and assist the user in the design of fuzzy queries, an automatic learning process based on the Genetic Algorithm-Programming (GA-P) paradigm [7] will be proposed in this paper.

In the following Sections, the basis of FIR and the composition of the proposed process are introduced. A preliminary experimentation will be also developed and some future research lines will be mentioned.

2 Fuzzy Information Retrieval

An IRS is constituted by three main components:

1. A *documentary data base*, which stores the documents and the representation of their information contents (usually based on index terms for textual documents). The *indexer module*, which automatically generates for each document its representation by extracting the document contents.

   In FIR, document representations and terms become fuzzy sets defined in the universe of terms and documents respectively, thus introducing a degree of relevance (aboutness) between a document and a term.

2. A *query subsystem*, which allows the users to formulate their queries and presents the relevant doc-

uments to them retrieved by the system. To do so, it includes a *query language*, that collects the rules to generate legitimate queries, and procedures to select the relevant documents.

FIRSs consider numeric or linguistic weigths in the query with different semantics [1], thus allowing the user to quantify the "subjective importance" of the selection requirements.

3. A *matching or evaluation mechanism*, which evaluates the degree to which the document representations satisfy the requirements expressed in the query and retrieves those documents that are judged to be relevant to it.

   A degree of document relevance is introduced in FIR —the so called *retrieval status value* (RSV)— by considering fuzzy operators appropriately performing the matching of queries to documents in a way that preserves the semantics of the former.

Let D be a set of documents and T be a set of unique and significant terms existing in them. The indexer module of the FIRS defines an indexing function:

$$F : D \times T \rightarrow [0,1]$$

In this paper, we use the normalized *inverted document frequency* [11]:

$$w_{d,t} = f_{d,t} \cdot log(N/N_t) \qquad ; \qquad F(d,t) = \frac{w_{d,t}}{Max_d\, w_{d,t}}$$

By projecting this "aboutness" fuzzy relation, a fuzzy set can be associated to each document and term:

$$d_i = \{< t, \mu_{d_i}(t) > \mid t \in T\} \quad ; \quad \mu_{d_i}(t) = F(d_i, t)$$

$$t_j = \{< d, \mu_{t_j}(d) > \mid d \in D\} \quad ; \quad \mu_{t_j}(d) = F(d, t_j)$$

where $f_{d,t}$ is the frequency of term t in document d, N is the number of documents in the collection and N_t is the number of documents containing term t.

Once the RSV of each document is known, the query subsystem affords a fuzzy set q specifying the degree of relevance of each document to the processed query:

$$q = \{< d, \mu_q(d) > \mid d \in D\} \quad ; \quad \mu_q(d) = RSV_q(d)$$

Thus, documents can be ranked in order of relevance in the query response. The user can specify the final relevant document set in two different ways: providing an upper bound for the number of retrieved documents or defining a threshold σ for the relevance degree.

The matching mechanism operates in a different way depending on the interpretation associated to the numeric weights included in the query (see [1, 3]). In this contribution, we consider the *importance* interpretation, where the weights represent the relative importance of each term in the query.

In this case, the RSV of each document to a fuzzy query q is computed as follows [12]. To maintain the semantics of the query, the term weighting has to take a different form according as the single term queries are ANDed or ORed. Therefore, assuming that A is a fuzzy term with assigned weight w, the following expressions are applied to obtain the fuzzy set associated to the weighted single term queries A_w (in the case of *disjunctive queries*) and A^w (for *conjunctive ones*):

$$A_w = \{< d, \mu_{A_w}(d) > \mid d \in D\}$$

$$\mu_{A_w}(d) = Min\ (w, \mu_A(d))$$

$$A^w = \{< d, \mu_{A^w}(d) > \mid d \in D\}$$

$$\mu_{A^w}(d) = Max\ (1 - w, \mu_A(d))$$

On the other hand, if the term is negated in the query, a negation function is applied:

$$\overline{A} = \{< d, \mu_{\overline{A}}(d) > \mid d \in D\} \quad ; \quad \mu_{\overline{A}}(d) = 1 - \mu_A(d)$$

The single fuzzy set representing the RSV of the compound query is obtained by combining the weighted term fuzzy sets by means of the following operators:

$$A\ AND\ B = \{< d, \mu_{A\ AND\ B}(d) > \mid d \in D\}$$

$$\mu_{A\ AND\ B}(d) = Min(\mu_A(d), \mu_B(d))$$

$$A\ OR\ B = \{< d, \mu_{A\ OR\ B}(d) > \mid d \in D\}$$

$$\mu_{A\ OR\ B}(d) = Max(\mu_A(d), \mu_B(d))$$

3 A GA-P Algorithm to Learn Fuzzy Queries

Genetic Algorithms (GAs) [10] have been succesfully applied as a tool to assist the user in the query formulation process (see [5] for a review including this and other different applications of GAs to IR). One of the approaches is based on automatically generating the query best describing the user's needs —represented in the form of an initial set of relevant (and optionally non relevant) documents— by means of an off-line process in which no user interaction is required. This operation mode is included in the usual *machine learning paradigm* and has been called *inductive query by example* (IQBE) by Chen in [2].

In [9], Kraft et al. proposed an automatic process of this kind to learn the whole composition of extended boolean queries (terms, weights and logical operators) for an FIRS. It is based on a variant of GAs, Genetic Programming (GP) [8], which evolves structures encoding programs such us expression trees. GP algorithms perform really well in the generation of structures, which are adapted both by crossover and mutation, but behaves bad in the learning of numerical

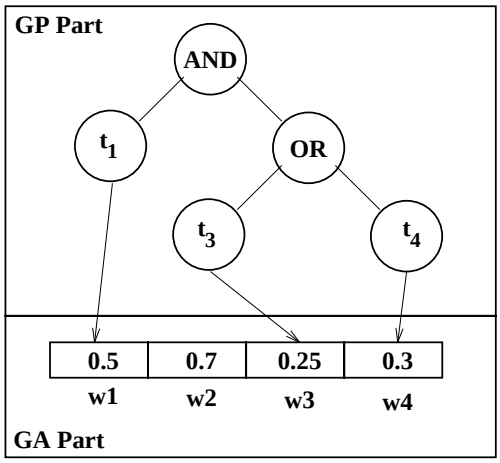

Figure 1: GA-P individual representing the fuzzy query 0.5 t_1 AND (0.25 t_3 OR 0.3 t_4)

values, which are only adapted by mutation. This is of significant importance in our case because these values corresponds to the query term weights.

The Genetic Algorithm-Programming (GA-P) [7], based on combining the traditional GAs with the GP technique to evolve complex expressions capable of handling numeric and symbolic data, can solve this problem. Each population member consists of both a value string and an expression as it is shown in Figure 3. The GP part of the GA-P evolves the expression (the query composition —terms and logical operators— in our case), while the GA part concurrently evolves the coefficients (the term weights). This is why our aim is to extend Kraft et al.'s algorithm to adequatelly deal with the term weights of the fuzzy query by considering a GA-P algorithm (for other applications of GA-P to Fuzzy Systems see [4, 13]).

The GA-P and GP make selection and child generation similarly, except that in the GA-P, crossover and mutation take place independently for the coefficient string and the expression component. Mutation and crossover rates for the coefficient string (using traditional GA methods) are independent from the rates for the expression part (using standard GP methods).

In our algorithm, the selection is steady-state [10]. Hence, each generation involves selecting two parents at random with a probability that grows with his adaption level —we consider the proportional scheme and the universal stochastic procedure—, crossing and mutating their GA and GP parts with probabilities $\{P_c^{GA}, P_m^{GA}\}$ and $\{P_c^{GP}, P_m^{GP}\}$ respectively, and introducing them in the current population only if they are better adapted than the worst two individuals.

To adapt the GP part, the usual GP crossover is considered [8], which randomly selects one edge in each parent and exchanges both subtrees from these edges between the both parents. Mutation is performed by a random selection of an edge and a random generation of a new subtree that substitutes the old one.

The GA part has a real coding issue and is adapted by the *Max-min-arithmetical crossover* [6] and the *Michalewicz's non-uniform mutation operator* [10]. Since the former works by generating four different offspring and selecting the best two ones, we combine the four GA parts with the two GP parts resulting from the GP crossover, thus obtaining two different groups of four descendants. The best from each group is the offspring finally considered.

The *generation of the initial population* is performed by computing random values in [0,1] for the GA part and random trees representing queries with a maximum predefined length and composed of randomly selected terms existing in the initial relevant documents provided by the user, for the GP part.

For the definition of the *fitness function*, two different possibilities are considered based on the classical precision and recall measures [9]:

$$F_1 = \frac{\sum_d r_d \cdot f_d}{\sum_d r_d} \quad ; \quad F_2 = \alpha \cdot \frac{\sum_d r_d \cdot f_d}{\sum_d r_d} + \beta \cdot \frac{\sum_d r_d \cdot f_d}{\sum_d f_d}$$

with $r_d \in \{0,1\}$ being the relevance of document d and $f_d \in \{0,1\}$ being the retrieval of document d in the processing of the current query. Simplest queries are preferred when having the same fitness value.

4 Experiments Developed

We have followed a similar experimental methodology to [9]. A documentary base composed of 359 abstracts taken from the *Library and Information Science Abstracts (LISA)* data base have been automatically indexed, obtaining a total number of 2609 different indexing terms. A user has selected a set of 82 relevant documents that has been provided to the system, which has been run considering the two different said fitness functions —with different values for α and β in F_2— and the use or not of the NOT operator.

The results obtained are shown in Tables 2 and 3, while the common parameter values are collected in Table 1. In the result tables, NOT stands for the consideration of the NOT operator, $\#rt$ for the number of documents retrieved by the learnt query, $\#rr$ for the number of relevant documents retrieved, R for recall, P for precision, and Sz for the generated query size.

In view of them, the process is not performing suitably. This fact is due to a main reason: *the composition of the fitness functions is not appropriated to represent*

Table 1: Common parameter values considered

Parameter	Decision
Population size	30
Number of generations	10000 (steady state)
GA Cross. probability	0.6
GA Mutation probability	0.1
GP Cross. probability	0.95
GP Mutation probability	0.05
Expression part limited to	15 nodes
Retrieval threshold σ	0.5

Table 2: Results obtained with fitness function F_1

NOT	$\#rt/\#rr$	R	P	Sz
N	359/82	1.0000	0.2284	3
Y	356/82	1.0000	0.2303	1

Table 3: Results obtained with fitness function F_2

(α, β)	NOT	$\#rt/\#rr$	R	P	Sz
(0.5,0.5)	N	359/82	1.0000	0.2284	3
(0.5,0.5)	Y	349/82	1.0000	0.2349	9
(0.5,1.0)	N	5/5	**0.0610**	**1.0000**	11
(0.5,1.0)	Y	1/1	0.0122	1.0000	1
(1.0,0.5)	N	359/82	1.0000	0.2284	3
(1.0,0.5)	Y	352/82	1.0000	0.2330	5
(0.5,0.75)	N	4/4	0.0488	1.0000	5
(0.5,0.75)	Y	1/1	0.0122	1.0000	1
(0.75,0.5)	N	359/82	1.0000	0.2284	3
(0.75,0.5)	Y	342/82	**1.0000**	**0.2398**	9

the problem goal. Focusing on F_1, we obtain two optimal solutions, both being very simple queries with total recall. Anyway, they are not useful due the large quantity of documents retrieved, goal not included in the fitness function guiding the search.

On the other hand, F_2 seemed to be able to solve this problem, because it considers two different criteria representing both problem goals. The problem is the weighted combination of the two criteria, which biases the search to one of the extremes of the pareto set of optimal solutions, thus causing a premature convergence of the GA-P algorithm. As seen in Table 3, solutions in an opposite part of the pareto are obtained when one of the weights dominates the other one, but the algorithm is not able to generate solutions with and appropriate balance between both criteria.

Multi-objective Evolutionary Algorithms can solve this problem since they look for compromise solutions satisfying all the criteria to different degrees. They also provide the user with a set of solutions differently distributed in the pareto, i.e., a set of candidate queries representing the user's needs in our problem. This will be our main research line in a very short future.

5 Concluding Remarks

A GA-P algorithm has been proposed to assist the user in the design of fuzzy queries for FIRSs. The process automatically generates complete extended boolean queries composed by terms, boolean operators and real-valued weights, describing the user's needs collected in a set of relevant documents provided by him.

The system did not behave properly in the experiments because of the wrong composition of the fitness functions, which biased the search to a specific zone of the pareto set. To solve this, multiobjective Evolutionary Algorithms will be considered in a future research.

References

[1] G. Bordogna, P. Carrara, G. Pasi, Fuzzy Approaches to Extend Boolean Information Retrieval, in: P. Bosc, J. Kacprzyk, Fuzziness in DataBase Management Systems (1995) 231-274.

[2] H. Chen, A Machine Learning Approach to Inductive Query by Examples: An Experiment Using Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing, Journal of the American Society for Information Science 49:8 (1998) 693-705.

[3] V. Cross, Fuzzy Information Retrieval, Journal of Intelligent Information Systems 3 (1994) 29-56.

[4] O. Cordón, F. Herrera, L. Sánchez, Solving Electrical Distribution Problems Using Hybrid Evolutionary Data Analysis Techniques, Applied Intelligence 10:1 (1999) 5-24.

[5] O. Cordón, F. Moya, M.C. Zarco, A Brief Study on the Application of Genetic Algorithms to Information Retrieval (in spanish), Proc. 4th International Society for Knowledge Organization (ISKO) Conference (EOCONSID'99), Granada, Spain (april, 1999).

[6] F. Herrera, M. Lozano, J.L. Verdegay, Fuzzy Connectives Based Crossover Operators to Model Genetic Algorithms Population Diversity, Fuzzy Sets and Systems 92:1 (1997) 21-30.

[7] L. Howard, D. D'Angelo, The GA-P: a Genetic Algorithm and Genetic Programming Hybrid, IEEE Expert (1995) 11-15.

[8] J. Koza, Genetic Programming. On the Programming of Computers by Means of Natural Selection, The MIT Press (1992).

[9] D.H. Kraft, F.E. Petry, B.P. Buckles, T. Sadasivan, Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback, in: E. Sanchez, T. Shibata, L.A. Zadeh, Genetic Algorithms and Fuzzy Logic Systems (1997) 155-173.

[10] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag (1996).

[11] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill (1989).

[12] E. Sanchez, Importance in Knowledge Systems, Information Systems 14:6 (1989) 455-464.

[13] L. Sánchez, Interval-valued GA-P Algorithms, IEEE Trans. Evolutionary Computation (1999), to appear.