Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

A MODEL FOR INFORMATION RETRIEVAL SYSTEMS*

PERFECTO DIPOTET

In this paper we propose a heuristic model for simple information retrieval systems based on classification. We consider that the model may be applied, mainly, in those cases in which the classification process is influenced by subjectivity. In the model we have employed some ideas proposed by Salton [1] for the retrieval model, by McQueen and Rettemayer for the partitioning technique and suggestions given by Benczur in the self-organization policy. The philosophy of this paper is that a heuristic approach can 6e used in a formal, to some extent, way in order to obtain good solutions for real life problems.

1. Classification process. Next we introduce some important definitions about the classification process.

Definition. *Let a set P, composed of elements $e \in P$, be given. These elements $e \in P$, possess some attributes a, inherent to them. We denote by $A=\{a\}$ the set of existing attributes a in P.*

Definition. *A descriptor d is a subset of A, $d \subset A$. We denote $D=\{d\}$ a set of defined descriptors. Let M be the number of different descriptors in D.*

An element $e \in P$ can be described by a string, which is a subset of descriptors $d_{i1}, d_{i2}, \ldots, d_{im}$, with $i_1, i_2, \ldots, i_m \in I=\{1, 2, \ldots, M\}$ and pairwise unequal indices.

The minimum $m_{\min}$ and maximum $m_{\max}$ number of descriptors allowed in the description of an element $e \in P$ are given.

Definition. *We denote E of the permissible descriptions of elements $e \in P$. This means that E is a classification language.*

Definition. *Let Y be a function from the space of existing elements P into E. We call Y the classification function. The value $Y(e)$ is called the description of element $e \in P$. $Y(e)$ takes the shape of the already mentioned subset of descriptors (string).*

The mapping $Y: P \to E$ defines an equivalence relation on P, where a given element e_i is equivalent to e_j if and only if $Y(e_i)=Y(e_j)$.

Definition. *We denote by $C=(Y(e), e)$ a classified element or element record or simply a record. We denote by $C=\{c\}$ a set of existing classified elements.*

Definition. *A request term r, at a time t, is a string, subset of descriptors, $d_{j1}, \ldots, d_{jm}$; where $j_1, \ldots, j_m \in I$, $I=\{1, \ldots, M\}$ and pairwise unequal. r retrieves a classified element with description $Y(e)$, if $Y(e) \supset r$. The series of request terms in a time period produce the reference strings.*

---

* Delivered at the Conference on Systems for Information Servicing of Professionally Linked Computer Users, May 23-29, 1977, Varna.

SERDICA Bulgaricae mathematicae publicationes. Vol. 10, 1934, p. 148—152.

Definition. *We denote by r_{ai}, the i-th record retrieved by a requestr at a time t. We denote by $R_a=\{r_a\}$ the set of records retrieved by a request r at a time t.*

2. Information storage. Next we determine the structure and the algorithms for the information storage system. We suppose C to be given. Records are sorted in order to be retrieved. C is generally formed by quite a lot of records and it is not possible, from operational point of view, to search sequentially requested records through the whole C. This situation can be avoided using some partitioning technique for C, according to similarity grouping ideas and some heuristic criteria.

Partitioning Technique. The partition of C is obviously given by:

$$C=\bigcup_{i=1}^{N} P_i \qquad P_i \text{ is the } i\text{-th partition of } C;\ N \text{ is the number of partitions.}$$

$$P_j \cap P_i=\emptyset \qquad \text{with } i, j \in I=\{1, \ldots, N\} \text{ for } i \neq j.$$

This means that there are not records in two or more partitions, and each record belongs to one partition. In the partitioning technique we must determine the expected maximum number of partitions, and the criteria to sort records in partitions.

We have denoted by N the number of partitions. We denote by T the technological characteristics of computer system configuration. It is obvious that: $N=F(C, (Y(e), e), T)$.

If we suppose records to be uniformly distributed among partitions and we denote by N_c and L, the expected maximum number of records in C and the expected maximum number of records in a partition, respectively: we can, roughly, expect that: $N=Nc/L$. L depends on records length, demanded access time to records in the actual application, and technological characteristics of the available configuration.

We assumed records to be uniformly distributed among partitions. We denote by q the expected number of partitions where one descriptor is allowed to occur. We denote by M the maximum number of descriptors in the system and M_p the expected maximum number of descriptors within a partition. It seems reasonable that: $M_p=Mq/N$.

Criteria to Store Records in Partitions. We denote by U_p user preferences at a given time t. We know that $r \in (U_p)$. Then at a given time t it is possible to assign different degrees of relevance, according to users preferences in C. As records are described, it is then possible to measure the degree of relevance (weight) of descriptors. Next we present a weighting procedure for the case when we are setting up the information storage system:

a) With the available information about P, select, a priori, the possible N partitions.

b) Take a random sample of W records. $W=N/4S^2$ with S^2 the, a posteriori, variance for the a priori assumed proportion of records belonging to each partition.

c) Store context related records in the same partition (later on we will use the concept of distance to store records in partitions).

d) On each partition we count n_i, $i \in I=\{1, \ldots, M_p\}$ with n_i denoting the number of occurrences of the i-th descriptor in the respective partition.

e) We sort descriptors within a given partition using the number of their respective occurrences for key.

f) With these M_p sorted descriptors we form an M_p component vector.

g) Coded weight of one descriptor within a given partition is the number corresponding to its respective ordered position in the M_p sorted descriptors.

All our operational work depends on elements description $Y(e)$, which we use to code in order to store them in memories. For coding $Y(e)$ we use the following procedure:

h) In the M_p components vector given in f, we assign value I to components (ordered positions corresponding to the M_p sorted descriptors) corresponding to descriptors present in $Y(e)$ and value $\varnothing$ to descriptors not present in it. In this way, the code for $Y(e)$ will be some kind of sparse binary vector.

i) The weight of $Y(e)$ will be given by the sum of the weights of the descriptors present in it.

A good feature of this coding procedure is the availability of both descriptors and elements weights simultaneously. Another important feature is that to change all records weights, if necessary, in a given partition, we need only to permutate some columns in the matrix formed by $Y(e)$ codes within the respective partition.

Definition. *We denote by* $\overrightarrow{D(e)}$ *a coded record in a given partition. We define the distance* θ_i *between* $Y(e_i)$ *and* $Y(e_j)$ *by the expression:*

$$\theta_{ij} = \sum_{k=1}^{M_p} \| \overrightarrow{Y(e_i)}_k | - | \overrightarrow{Y(e_j)}_k | , \quad \text{with} \quad e_i, e_j \in P$$

and $Y(e)_k$ *is the k-th component of* $\overrightarrow{Y(e)}$ *within a given partition we denote by* $\hat{\theta}$ *and* θ_R *the maximum possible distance between two records and between one record and a reference one, respectively.*

3. Retrieval and update algorithms. A retrieval system $\rho = (X, Z)$ has two parts. One is a representation function (information storage) X; the other is a retrieval algorithm Z. Given a request r for classified elements or records, the algorithm Z generated a sequence of processes which finally gives us back the adequate R_a. Z gives us also the statistical parameters for updating the retrieval system and provides a feedback from users to the classification process.

A retrieval system with update (X, Z, U) is a retrieval system (X, Z) with an updating algorithm U. Given the statistical information O about the performance of the retrieval system and given X, U generate a sequence of processes which perform the corresponding modification of the information storage structure.

We have already given a procedure which allows us the setting up of the information storage system. We have also given the weighting and coding procedures for $Y(e)$ and descriptors d_i. But in that weighting procedure we have used for parameter the number of occurrences of descriptors in the partition. Now we can improve this criterior using for parameters: the number of references to partitions; the number of records in partitions N_e and, for weighting descriptors, the number of references to each descriptor n_p. In this way it is also possible to obtain the most requested record in a time interval At, and to take it as a reference record for the affected partition.

Self-organizing Policy. As we have already mentioned $U_p = \varepsilon(t)$. For this reason it is necessary to examine periodically the accumulated of time where partitions are examined.

$$r = F(N_p \text{ or } \Delta t),\ \Delta t \text{ is a fixed time interval.}$$

This logical takes care of both instantaneous perturbations (noise) and of significant, but slow changing, information. At observation points we check N_p, U_p, $\hat{\theta}$ or θ_r and N_e.

We define by target values for a given partition the values given by $\hat{\theta} \leq \varepsilon$ or $\theta_R \leq \varepsilon$, in which ε is a fixed value.

A state for one partition, at a given observation point, is one of the possible combinations between two factors: change in reference record and partition overflow respectively.

Possible states are:

— actual reference record coincide with the former one (the one given in the former observation point) and partition does not overflow ($N_e < L$);

— actual reference record does not coincide with the former one and partition does not overflow;

— actual reference record coincide with the former one and partition overflows ($N_e > L$);

— actual reference record does not coincide with the former one and partition overflows.

It is obvious that both factors affect θ_R.

An action B, affecting the state of a partition at a given observation point, is given by a reduction of N_e or by changes in descriptor weights (or both). These actions are always looking for the most demanded record at a given interval of time. At and/or for the reduction of θ_R. Possible actions are:

— change descriptors weight and, consequently, records weight in a given partition;

— take a selected group of records off from the partition;

— change descriptors and records weights and take a selected group of records off from the partition;

— to keep the same state.

A self-organization policy π, in our case, is the prescription we are going to use for taking actions at observation points, until we reach the target value or until the point when we take the decision of stopping observations. Our policy π is a deterministic one, because there is an exact mapping between states and actions. In our case, the motion from one state to another depends (only) on the actual state and the action taken.

To observe partitions and to take actions imply some cost ω_c. It is necessary to take care that expected benefits of actions could be greater than the expected total cost, ω_c, of implementing them. It is obvious that here is present a loss function which is necessary to minimize.

Complete Update. When a partition reach its target value $\theta_k \leq \varepsilon$, we say that there exists a subset of C which is strongly demanded by users. In other words, when we request certain records, we retrieve a block of them. The information is now given by the block and not by isolated records. It seems reasonable to put the records, from this partition, into another operational environment where information is given by big blocks of records.

When partitions are only seldom referenced, after some elapsed time, we say that there exists a subset of C which is rarely demanded by users. In other words the benefits of the stored information is much lower than the cost of keeping it in the actual operational environment. It seems also reasonable, in some cases, to put the affected partitions in a less expensive operational medium.

Using the statistical information about user demands, is quite possible the exploitation of the informational potentialities of the retrieval system, at least for simple cases which are present very often in real life.

4. Conclusions. A test for a version of the here proposed model, applied to a document retrieval system, was implemented on a *CDC* 3300 computer. The parameters of the test were:

N_c	550 documents
M	200 descriptors
M_p	48 descriptors
N	8 partitions
M_{min}	3 descriptors
M_{max}	6 descriptors
L_{min}	40 documents

The test brought very good results in recall and precision factors adjustable changing the value. Intentionally wrong classified records were detected and the accumulation of statistical parameters, in more than 200 requests, has worked very good. The difficulty present was in the setting up of the information storage system because it was needed to store manually so many records per partition to obtain stability in the weighting procedure. The data accumulated in the test was enough to apply the complete update criterion for a case of strongly demanded subset of records.

In the near future by the same author will be published a paper facing the same problem, but using the formal approach given by similarity grouping methods [2]. It will then be possible to compare the advantages and drawbacks of both models.

REFERENCES

1. G. Salton. Automatic information organization and retrieval. New York, 1968.
2. M. R. Anderberg. Cluster Analysis for Application. New Yotk, 1973.

Cuban Academy of Sciences
IMACC Havana

Received 13. 9. 1977