87-MMD

Mathematical Modelling in Science and Technology

THE FIFTH INTERNATIONAL CONFERENCE
Berkeley, California, USA July 1985

Edited by
Xavier J.R. Avula
University of Missouri-Rolla
George Leitmann
University of California, Berkeley
C.D. Mote, Jr.
University of California, Berkeley
Ervin Y. Rodin
Washington University

PERGAMON JOURNALS LIMITED
A Member of the BPCC PLC Group of Companies

87-MMD.

A METHOD OF MATCHING DATA

Naphtali Rishe[1]

Michael Hanani[2]

[1] Computer Science Department, University of California, Santa Barbara, Ca 93106, USA.

[2] Center for Health Studies, Ben-Gurion University, P.O. Box 653, Beer Sheva 60821, Israel

Abstract. A probabilistic model and a software implementation have been developed to aid in finding missing persons and in related applications. The method can be applied generally to find most probable correspondences between two sets of imprecisely described objects. These can be descriptions of illnesses *vs* patients (diagnoses), job offerings *vs* job applicants, special tasks *vs* a personnel file (task assignment problem), *etc.* Every object of the two sets is described by a collection of data, a significant part of which can be erroneous, unreliable, imprecise or given in several contradicting versions. Among the parameters of the method is the following information about each of the data item types and of some of their possible combinations (the parametric information does not depend on the actual data): its logical characteristics, its importance relatively to other types of data items, the meaning and the relative degrees of kinship between values of this data item for two objects to be compared (*e.g.* kinship of equal values; phonetic kinship; numeric kinship, whose degree is proportional to the inverse of arithmetic difference between the values; matrix of kinship degrees defined for possible pairs of values), interpretation of multiplicity of values for this data item for one object, the *a priori* probability of data item's correctness (in addition, the probability of any value for any object can provided in a set of objects' descriptions by an investigator who gathers the actual data), *etc.*

A straightforward implementation of the method by software would result in unfeasible time complexity for large sets of objects. Therefore special algorithms have been designed to preprocess the sets of descriptions so that the time of matching-finding is reduced by an order of magnitude while the probabilistic output remains unaltered.

Keywords. Matching; unreliable data.

PURPOSE

This paper is a report on a software system DMS, which has been developed to assist in finding correspondences between the objects of two large sets (populations).

The following is the general purpose of DMS (Data Matching System).

Two files are given, A and B, each of which is a set of logical records. Each record is a collection of data about an object. The data is presumed to have been collected by unreliable processes, which have caused impreciseness, errors, and omissions. The data may have been collected manually. Conflicting data may have been supplied by different witnesses.

There is a correspondence between some A-objects and some B-objects. A particular case of such correspondence is the identity of two objects represented or described by possibly unidentical records of the two files. (An object x can be described by a record $a \in A$ and by a record $b \in B$, where $a \neq b$.) One A-record can correspond to one B-record, to many B-records, or to no B-records.

The correspondence might be found by an expert, who, given unlimited time, would analyze every pair of A and B records. However, when there are many records, a mechanized process is needed in order to reduce the expert's search space.

For example, if each file contains 1000 records, then an expert unassisted by a computer would have to analyze 1000 possibilities per every A-record, and this may be infeasible. The manual intelligent work would be reduced 100 times if a computer program could suggest about 10 candidate B-records per every A-record, in order to narrow down the search space. The selected candidates must include all the B-records which may correspond to a given A-record, if there are any.

The computer program cannot substitute the expert, since the decision on the correspondence might be a highly intelligent one. However it should filter out all those B-records which are highly unlikely to correspond to the given A-record. The selected candidates should be sorted from the most probable to the most improbable, and rough degrees of probability should be assigned to the candidates.

SOME USES OF DMS

In addition to the current application of DMS, it can be used for the following applications:

1. Search for relatives. The file A will contain requests to find lost relatives. The file B will contain data on all the residents of a country. Two records will match if they describe the same person, even though the descriptions were made in different ways, at different periods in the person's life, and probably with many errors.
2. Assistance in medical diagnosing or in control over diagnoses. The file A will contain descriptions of patients, and the file B -- descriptions of diseases. A patient's record will match a disease's record if it is probable that the patient has the disease.
3. Assistance for a marriage match-maker or a dating service. Here there are two applications of different complexity.

a. A and B are file/s of available singles. Two records match if the described persons may be compatible.

b. A is a file of descriptions of wanted (imagined) spouses or dates; B is a file of available singles. A fantasy description is matched by the closest descriptions of real persons.

The first case requires a sophisticated definition of the matching criteria, which is an input of DMS.

4. Search for wanted merchandise (described in A) amongst available merchandise (catalogued in B.)

5. Matching between descriptions of unsolved crimes and a file of known criminals.

For most of the aforementioned applications specialized systems exist besides DMS. The purpose of DMS was to provide an efficient and reliable *application-independent* solution for all the above and many other applications. An input parameter to this general system is a definition of an application, *i.e.* the logical criteria of matching.

THE INPUT FILES

Every file consists of logical records. Every record has, *inter alia*, fields which may be used for the computerized matching. Every field in a record may contain a value, or several values (occurrences), or no value at all. When there is no value in the field, still a value may exist in the real world, but it has not been reported. When there are several values, their multiplicity may have several different interpretations, *e.g.* the following:

a. In the real world there is only one value for this field for this object, but this value was not known when the record was reported. Instead, several hypotheses or contradicting evidences were recorded.

b. This field may indeed have several values simultaneously for one object in the real world.

Every value in a field may be accompanied by a degree of its reliability estimated by the reporter of the value. This degree is reported when it is different from the default general estimate of reliability of values of this field.

For example the field *headache strength* in a file of patients' descriptions may have a default degree of reliability 0.8. For a hypochondriac this degree may be reported as 0.5. If the value was obtained by an instrument measurement, rather than from patient's words, the degree of reliability may be reported as 0.9. The reliability degree of corresponding field in the file of disease descriptions may have a default of 0.5. For a particular disease, for which the headache is a primary symptom, the reliability degree of 0.9 may be reported. For a disease which may occur without headache at all, this degree may be reported as 0.1.

THE META-DATA DICTIONARY

The system determines the logic of data matching according to meta-data (criteria and other general information) supplied to it in the input. The meta-data are kept in a data base (a dictionary), and may also be modified for any particular run of the system.

For every field type meaningful for the matching process the dictionary contains the following information:

1. A definition of the degrees of comparability between values, *i.e.* the possibilities to compare this field's values in the file A to those in B. This degree is a number between -1 and 1, and it can be defined in one of the following ways.

   a. Identity of the compared values. (The degree of comparability of v_1 and v_2 is 1 if $v_1 = v_2$, and -1 otherwise.)

   b. Lexicographic approximation.

   c. Numerical approximation.

   d. Phonetic approximation.

   e. Discrete comparison: a table is given of pairs of comparable values and their degrees of comparability.

   When two values of a field in two records of the two files are compared, the degree of their comparability is produced according to one of the above criteria.

   When the two values cannot be compared, their degree of comparability is -1. This, however, does not mean that the two records cannot match, since some values may be erroneous: the decision on matching is made taking into account all the fields and their values.

2. The meaning of multiple occurrences of values in the field, if this is possible. (The major possibilities were described in the previous section.)

3. The weight of importance of comparability in this field with respect to other fields.

4. The weight of negative importance of incomparability in this field with respect to other fields.

5. Default reliability degrees of the field for each file.

6. A degree of security that no spelling/punching/communication-channel mistakes can appear in the values of the field. (This is distinguished from a mistake in report of facts, i.e. false information.)

7. Dependencies between different fields. (Mathematical formulas to compute derived fields, logical connections between values of different fields.)

THE LOGICAL PRINCIPLE OF MATCHING

The essence of the logic of matching can be roughly summarized by the following procedure. (The actual algorithm is quite different due to efficiency-optimization considerations and also due to treatment of irregular criteria.)

For every A-record a and every B-record b the degree of keenship between a and b is:

$$\sum_{f \in field\text{-}types} \underset{v_f^A \in a.f}{\begin{bmatrix} \text{MAX} \\ or^* \\ \text{MEAN} \\ or^* \\ \text{WEIGHTING} \end{bmatrix}} \underset{v_f^B \in b.f}{\begin{bmatrix} \text{MAX} \\ or^* \\ \text{WEIGHTING} \end{bmatrix}} (d(v_f^A, v_f^B))$$

where $d(v_f^A, d_f^B)$ is the weighted degree of comparability defined as follows:

$$d(v_f^A, v_f^B) =$$

the degree of comparability of v_{fa} to v_{fb} $\times$
positive or negative weight of the field $\times$
the degree of reliability of v_{fa} $\times$
the degree of reliability of v_{fb} $\times$
adjustment by a nonlinear function of the degree of security of the field f.

* The appropriate statistical function (MAX / MEAN / WEIGHTING) is selected according to the meaning of the multiple occurrences of values in the field f, as given in the meta-data dictionary.

For every A-record a set of closest B-records is displayed with their degrees of kinship.

EFFICIENCY

If the above matching logic were implemented directly, the program would have to run too long, and perhaps would be infeasible for large files.

Here is an example. Assume that each file contains 1000 records, 100 fields, 3 occurrences per field in a record. Let x be the time to calculate the weighed degree of comparability between two values. Then the total time is

$$1000 \times 1000 \times 100 \times 3 \times 3 \times x = 9 \times 10^8 \times x$$

The physical algorithm of DMS is not a direct implementation of the above logic. The quadratic time is reduced to linear.

Although the linear algorithm employed produces output which is not identical to the output which would be produced by the above quadratic algorithm, they are interchangeable for any practical purpose.

If the quadratic algorithm would produce a set S_a of potential matches for a given A-record a, the linear algorithm will produce the same B-records S_a and may additionally produce a few extra B-records, $NOISE_a$. usually this $NOISE_a$ is empty, and in any case its expected cardinality is much less than the number of the candidate records given in S_a.

In any case the output is examined by a human expert to select real matches, if there are any. In the first case, after the quadratic algorithm, the set S_a is examined. In the second case, after the linear algorithm, the set $S \cup NOISE_A$ is examined, but the same matches are found since the matches are in S_a.

No potential match is omitted by the linear algorithm. The expectation of the work to be done by the human expert in analyzing the output may rise insignificantly ($\tilde{}1\%$).

The following are some of the steps done to reduce the time:

1. The files are preprocessed vertically and horizontally to leave only relevant data.

2. For every field a set of all different values actually appearing in the files is found, enumerated, and the values (which could be long strings) are substituted by their numbers in the set. Apart of reducing the length of the values, the enumeration facilitates direct access to comparability tables (in time O(1)) and other savings.

3 Further reduction of the set of values in the field is done by analyzing the equivalence relations among values.

4. The values of the fields in the file A are substituted by vectors or sparse vectors. The indexes of these vectors are the numbers assigned to the distinct values of the field which actually appear in file B. The entries in the vectors are degrees of comparability between the original value in A and the indexed values of B, modified by all the data known from the A-record and the meta-data-dictionary (weights, the degree of reliability, the degree of security.)

5 When in a field of an A-record there are several occurrences of values, the corresponding vectors are combined into one vector by vector operations dependent on the meaning of the multiple occurrences (as recorded in the meta-data dictionary.)

   For example, if the multiple occurrences mean different versions or hypotheses of the single occurrence in the real world (with probability estimates for every hypothesis), then the combined vector is obtained by entry-wise maximization of the former vectors:

   $$VEC_{combined}(i) = \mathrm{MAX}(VEC_1(i), VEC_2(i), \ldots)$$

6. When for a given field very many distinct values appear in B and approximation is defined by some special algorithms, e.g. the phonetic approximation, a different technique of hashing is used.

DISCUSSION OF OTHER APPROACHES

At the first glance, several simpler procedures might be considered for matching the files. However, these procedures would not produce the desired results. Here are two of these procedures:

1. A human expert is equipped with a powerful query language facility. While working on one A-record the expert formulates queries to retrieve B-records. If the output is not satisfactory, the expert reformulates the query in a trial-and-error process.

2. For every A-record a query is automaticly generated to retrieve B-records.

Both procedures fail to take into account the low reliability of the data.

Consider the following example. Let every record consist of 100 fields. Consider one A-record (For simplicity let us ignore multiple occurrences per field, missing values, and probability estimates for occurrences):

$$F_1^A : v_1,\ F_2^A : v_2,\ \cdots\ ,\ F_{100}^A : v_{100}$$

Assume that for every field one can formulate a suitable condition to be incorporated in the query. *E.g.*, if the approximation in F_1 is numerical, a condition like

$$cond_1 : [F_1^B = v_1^A \pm 10\%]$$

could be considered. Now, how can one compose a query from these conditions? If their conjunction is taken:

$$query = cond_1 \wedge cond_2 \cdots \wedge cond_{100}$$

then no B-record will appear in the output, since it is almost certain that at least one of the 100 values has been reported erroneously in every potentially-matching B-record.

If the disjunction of the conditions constitutes the query:

$$query = cond_1 \vee con_2 \cdots \vee cond_{100}$$

then almost the whole file B will be output for the above A-record, and the major goal of narrowing the search space will not be achieved.