Similarity Detection among Data Files – A Machine Learning Approach

M. Dash & H. Liu
Department of Information Systems & Computer Science
National University of Singapore, Singapore 119260

Abstract

In any database, description files are essential to understand the data files in it. However, it is not uncommon that one is left with data files without any description file. An example is the aftermath of a system crash; other examples are related to security problems. Manual determination of the subject of a data file can be a difficult and tedious task particularly if files are look-alike. An example is a big survey database where data files that look alike are actually related to different subjects. Two data files on the same subject will probably have similar semantic structures of attributes. We detect the similarity between two attributes. Then we create clusters of attributes to compare the similarity of the subjects of two data files. And finally a machine learning technique is used to predict the subject of unseen data files.

1 Introduction

A database consists of many files some of which are created for description purpose. A data dictionary created for this purpose stores the schema, subschema, integrity rules, security information, and description of all data items in the database [1]. In other cases, a data file may simply be accompanied by a name file which typically contains descriptions of all attributes. These are helpful in understanding the data files in the database. Situations may arise where one is unfortunately left with only data files. The following lists a few such situations.

- **A System Crash Problem**

  System's crash occurs. When it happens the aftermath is unpredictable. Say you have many databases which also include schema designs or name files. After the system crashes you find that most of the description files are lost and you are unfortunately left with some data files without any description file. Manual determination of the subject of each data file is not an easy task particularly if many of them look alike as is usual in databases with many data sets.

  Imagine a situation where a big survey is conducted on various groups of people on different aspects (*e.g.*, career, lifestyle, diet habits). If the system crashes and some of the description files are lost then it is very difficult to determine the subjects of the data files manually as the survey files generally look alike.

- **An Intelligence Agency Problem**

  Another situation can be related to the intelligence agencies. Typically, intelligence agencies try to steal data from the counterparts and try to decode it to reveal important information.Say an intelligence agency somehow gets access to a few data files but is unable to find the corresponding description files. This may happen if the data is tapped while being communicated. These data files are not much of use unless the subject of each one is determined.

- **Comparing Information Without Leaking It**

  This problem was discussed in [4]. A good number of situations and solutions are mentioned in it. In our opinion creating a database file without any description file is a way of storing information without leaking it even if others have access to it. In such a case the problem of detecting similarity between data files is a relevant problem.

All the above cases and many other similar situations necessitate a way of determining the subject of a data file. This problem can be simplified by converting it to a test of similarity between two data files where the subject of one data file is known and that of the other is unknown. If the test says that both are similar then the subject of the unknown data file is the same as that of the known data file; otherwise they are of different subjects.

0-8186-8230-2/97 $10.00 © 1997 IEEE

To our knowledge similarity between two data files is not yet attempted. But substantial research (*e.g.*, [2], [5], [6]) has been done to find the similarity between attributes in two heterogeneous databases. The research specifically tries to integrate knowledge between heterogeneous databases by determining the attributes which are similar. For each attribute a set of features are extracted and then compared to determine the similarity. These features can be obtained from the schema design [10] or from the data contents [5], [7], [8], or both [6]. But as no description file is available we use techniques in [5], [7], [8], and [6] to extract a set features for each attribute. This idea is helpful in finding the similarity between two data files. The intuitive idea is that analysis of the attributes' values of a data file can reveal its semantic structure. If the semantic structures of two data files are found to be the same or nearly so, then they are similar; otherwise they are dissimilar. A payroll data file is different from a student grade data file due to the different semantic structures of their attributes. The payroll data files share some commonalities which are unique to them. These commonalities are possibly different from those of the student grade data files. We assume that all input data files are in the form of a sequence of records each of which is a row of attribute values without any missing value. This assumption, in our opinion, is not too restrictive as most of the data files are of this form. The method described in this paper will help the user in determining the similarity between two data files. The problem is suitably converted to use a typical machine learning technique, which helps tackle the variations of the problem domain.

The remainder of this paper is organized as follows. In the next Section we review the related work. In Section 3 we discuss the concept of similarity checking and describe our approach. The similarity detection algorithm is presented in Section 4. We describe a case study in Section 5. The paper concludes in Section 6 with discussions on various issues.

2 Similarity Checking

We use the techniques applied in [5], [7], [8], and [6] to extract a set features for each attribute. We broadly group the attributes as (a) unique or (b) non-unique. An attribute having a unique value for each instance is unique, but an attribute taking real or integer values may have unique value for each instance. Unique attributes like ID number may consist of only digits. Hence only those attributes taking real values are not checked for uniqueness; all other attributes are candidates for the uniqueness test. Any attribute having values consisting of a compulsory decimal point and no character other than digits and an optional sign (+/-) in the beginning followed by an optional currency symbol, and optional commas "," in between will be considered as real. The non-unique and non-real attributes are grouped as binary or integer (see Figure 1) depending on whether it has two values or more respectively. The nominal attributes having characters other than digits can be assigned integral values in many ways. We sort them in alphabetical order and assign contiguous values starting from 1. This method of value assignment to nominal attributes may create some problem in certain situations; but in our opinion, given no *a priori* information this method is quite robust.

Two different sets of features are extracted depending on whether an attribute is unique or not. Features extracted for unique attributes are: (a) maximum percentage of non-alphabetical characters, (b) maximum number of white space characters, and (c) statistics on length (maximum length and other features such as average length, variance of length and coefficient of variance). The following features are extracted for non-unique attributes (after assigning integral values to nominal attributes): (a) real or not, (b) integer or not, (c) binary or not, (d) average (arithmetic mean), (e) variance, (f) coefficient of variance. The coefficient of variance helps in detecting different units (*e.g.*, salary in thousands or not) and granularity levels (*e.g.*, salary per month or per week) of attributes.

The separation of attributes into two groups helps in detecting a major mismatch regarding the presence of unique fields. A typical survey file which does not disclose individuals' identities by not mentioning name, social security number, driving license number, etc. is totally different in this sense from a typical payroll data file which contains the employees' details that is unique. Hence similarity between two data files can be determined by finding the

- similarity among the unique attributes, and
- similarity among all other fields.

If one file contains unique fields and the other does not, then they are considered dissimilar. If both files do not contain any unique attribute, then the non-unique attributes are tested for similarity. Otherwise both groups of attributes are tested for similarity separately.

Two data files on the same subject will have attributes with similar semantic structures. Hence, intuitively, if one creates clusters of attributes based on the extracted features then for data files on the same

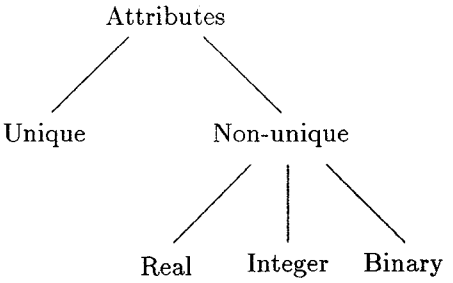

Figure 1: Classification of Attributes

subject the clusters will have less distance than that for dissimilar data files. We use group average (arithmetic mean) method [3], [9] for clustering. Minimum distance between two clusters is given by:

$$d(A, B) = min[d(u, v) \text{ for all u in A, v in B}].$$

User must provide the required number of clusters for each group of attributes. It is suggested that ideally the number of clusters should be equal to the number of distinct attributes [6]. In other words, all similar attributes in a data file will cluster together; for example, salary, gross salary, and net salary will probably cluster together in case of a payroll data file. Average correlation coefficients are calculated for the clusters of non-unique attributes. Sum of the correlation coefficients for each pair of attributes in each cluster is found out and it is divided by the total number of pairs of attributes in the cluster. Correlation coefficient helps in detecting the degree of interdependence among the attributes of a cluster. In case of a payroll data file, the gross salary and net salary are positively correlated most of the times. Hence, if both of these are in the same cluster then the corresponding correlation coefficient value will be high. This helps distinguish data files which have attributes with similar specifications but differently correlated. The correlation coefficient of each cluster is appended to the vector representing that cluster. For clusters having only one attribute, the correlation coefficient is set to 1.0.

3 Algorithm

We convert the problem of similarity detection to a machine learning problem. Machine learning algorithms are preferred in situations where the problem domain varies a lot. This is particularly suitable for our problem, as for example, a payroll data created for one company will have differences from a payroll data for another company.

Similarity Detection Algorithm: (Please refer to the block diagram in Figure 2.)

Threshold Setting Phase:

1. *Create or select a prototypical data file.*
2. *Find clusters for the prototypical data file.*

(a) Extract features from each attribute of the prototypical data file.

(b) Normalize each value to the range 0.0-1.0 for each feature.

(c) Separate the unique attributes from others.

(d) Find clusters of attributes in each group.

(e) Calculate average correlation coefficient of each cluster in the non-unique group and
append it to the vector representing the cluster.

(f) Name the two sets of clusters data as group1 and group2.

3. *Create a few manipulated data files.*
4. *Determine threshold distances.*

(a) Perform steps 2(a)-(e) to find clusters for each manipulated data file. The number of
clusters for each group is same as that for the prototypical data file.

(b) Calculate the minimum distance for each cluster in group1 and group2 from corresponding
the clusters of each manipulated data file.

(c) Set the threshold distance to the largest of these minimum distances for each

Similarity Detecting Phase:

5. *Detect similarity for an unseen input data file.*

(a) Perform steps 4(a)-(b) for the input data file.

(b) If any minimum distance is greater than the corresponding threshold distance then
reject the input as dissimilar; otherwise select it as similar.

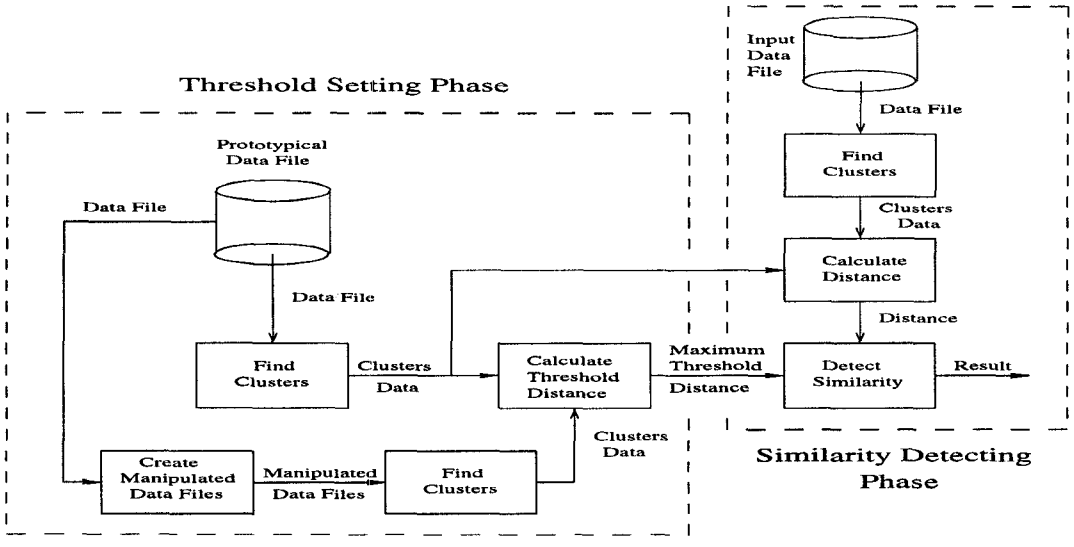

Figure 2: Block Diagram of Similarity Detection Algorithm

The algorithm consists of a threshold setting phase and a similarity detecting phase (see Figure 2). In the threshold setting phase a prototypical data file on the subject at hand is chosen. An easy way of creating one such is by including most of the important attributes on the subject. A prototypical student grade data file should contain some attributes for identification of a student such as name, student ID number, a few attributes for grades, percentages and class level, etc. Care must be taken to see that important attributes are not omitted. A few excess attributes will not hurt the algorithm's overall performance. Features are extracted for each attribute, unique attributes are separated, and clusters are found for each unique (if present) and non-unique groups.

A few manipulated data files are required to set the threshold distances used for detecting the similarity. These files can either be created from the prototypical data file or selected independently. Ideally each manipulated data file should be different in some sense from the others. The difference can be in the number of attributes, types of attributes, specifications of attributes, contents of attributes, etc. For example, a manipulated payroll data file may not include any attribute for allowances and/or deductions, it may include employee's address as an addition, the salary may be per week instead of per month. User's background knowledge of the subject is important in this step. Each manipulated data file should have the attributes important for the subject. But these attributes may not be exactly the same as that for the prototypical data file. Clusters are created for each manipulated data file the same way as that for the prototypical data file. The number of clusters for each group can be set in two ways. It may be set to the number of clusters for the corresponding groups of the prototypical data file. In the second way, the distance at which clustering was stopped for each group of the prototypical data file is recorded and it is used as a stopping criterion in the clustering of each manipulated data file. Minimum distance is calculated between the corresponding clusters in the prototypical and the manipulated data files. The largest of these minimum distances is the threshold distance for each cluster.

Given an unseen input data file, if the prototypical data file has unique attributes but the input data file does not, or vice versa, then reject the input data file as dissimilar. Find clusters and calculate the minimum distances from that of prototypical data file. Minimum distances are calculated for each cluster of the prototypical data file from the clusters of the input data file for each group separately. If any distance is found to be greater than the threshold distance then reject the input data file as dissimilar; otherwise accept it as similar.

4 Case Study

In this section we briefly describe a case study based on the payroll subject. We have chosen payroll as the subject as it is very common and widely used.

Step 1: A prototypical payroll data file was created as shown in Table 1. Values of attributes 1–4 are randomly generated with a uniform distribution. Attributes 5 and 6 are positively correlated. Values

of attribute 7 (date of birth) are generated using uniform distribution and values of attribute 8 (age) are found out from these. The following formulae are used for the other attributes: (a) Salary + Overtime + Perks = Gross_Salary, (b) CPF_Deduction + Insurance_Deduction + Tax_Deduction = Total_Deduction, and (c) Gross_Salary – Total_Deduction = Net_Salary. A total of 200 instances were generated.

Step 2: Attributes 1,2,3,4,7 are found to be unique although attribute 7 may not always be unique. All other attributes are non-unique. In both unique and non-unique groups, three clusters are created. This equality is a mere coincidence. A user should decide the number of clusters using his knowledge of the subject. Clusters of the unique group are: (a) {1,7}, (b) {2}, and (c) {3,4}; and the non-unique group are : (a) {5,6,8}, (b) {9,12,17}, and (c) {10,11,13-16}. Correlation coefficients of the clusters in the non-unique group are: 0.56 for cluster (a), 0.99 for cluster (b), and 0.01 for cluster (c).

Step 3: A few manipulated payroll data files are created from the prototypical data file by (i) varying the number of attributes, (ii) varying the field specifications, (iii) adding new attributes such as employee address, etc. Care is taken not to deviate much from a typical payroll data file.

Step 4: Clusters are created for each manipulated data file. Minimum distance of each cluster in the prototypical data file is calculated and threshold distances are set to the largest of these minimum distances. The threshold distances for clusters of unique group are (0.12, 0.14, 0.17) and for non-unique group are (0.19, 0.08, 0.10).

Step 5: Four unseen input data files consisting of two payroll data files and two different data files on subjects other than payroll are input. Due to space restriction we describe two of these four tests briefly: one payroll data file and one student grade data file. Attributes of the payroll data file are as shown in the Table 2.

Values of attributes 1–5 are generated as before. The following formulae are used for attributes 6–10: (a) Salary + Allowances = Gross_Salary, and (b) Gross_Salary – Deductions = Net_Salary. Attributes 1,2,3 are unique and other attributes (4–10) are non-unique. Clusters of the unique group are: (a) {1}, (b) {2}, (c) {3}; and the non-unique group are: (a) {4,5}, (b) {6,8,10}, (c) {7,9}. Correlation coefficients of the clusters in the non-unique group are 0.63 for cluster (a), 0.99 for cluster (b), and 0.025 for cluster (c). The minimum distances are (0.02, 0.01, 0.01) for unique clusters, and (0.14, 0.01, 0.04) for non-unique clusters. *As both sets of distances are well under the threshold distances, this file is selected as similar.*

Next a student grade data file was created as shown in Table 3. Values of attributes 1–6 are generated as in case of payroll data file. Attributes 9 and 10 are positively correlated. Attributes 1,2,3 are unique and attributes 4–11 are non-unique. Clusters of unique group are: (a) {1}, (b) {2}, (c) {3}; and non-unique group are: (a) {4,5,7,8,9,11}, (b) {6}, (c) {10}. Correlation coefficients of the clusters in the non-unique group are 0.03 for cluster (a), 1.00 for cluster (b) and 1.00 for cluster (c). The last two values are 1.00 as the corresponding clusters consist of only one attribute each. The minimum distances are (0.02, 0.01, 0.01) for unique clusters, and (0.20, 1.10, 2.02) for non-unique clusters. Note that the distances for clusters of the unique group are the same as those for the payroll input data file as the specifications of the attributes in both files are almost the same. The second vector of distances has values much higher than the threshold values, *hence, it is rejected as dissimilar.*

Table 4 shows the results for four input data files. This table does not show any result for data files having no unique attribute as these are rejected after finding no unique attribute.

5 Conclusions and Future Work

In this paper we have showed that the similarity between subjects of two data files without any description file can be checked. By our definition in this paper, two similar data files must be based on the same subject. Our method tries to find the similarity between two data files by comparing the semantic structures of attributes. It groups the attributes as unique or non-unique. A hierarchical clustering algorithm based on group average distance measure is used. A simple machine learning technique is used to set the threshold values. This is particularly helpful as data files on the same subject can vary in various dimensions. The experimental results show that our method is able to detect the similarity.

The following are some of the strong points of our method. It addresses a difficult problem and tries to solve it in a simple and feasible manner. A simple machine learning technique is used to handle the diversity of the problem domain. Unlike typical machine learning methods, the user is not required to set any threshold values. The threshold values are automatically set depending on the manipulated data files.

As in any machine learning method, user's interaction is important for our method to perform well. User should provide a good prototypical data file. This file should not exclude any important attribute. Hence

Serial No.	*Name of Attribute*	*Specification*
1.	Employee Number	3 digits, "-", 4 digits
2.	Employee Name	maximum 25 alphabets with 1/2 white space
3.	Social Security Number	3 digits, "-", 3 digits, "-", 3 digits
4.	Employee Phone	"(", 2 digits, ")", "-", 3 digits, "-", 4 digits
5.	Department Number	1 digit (1/2/3/4)
6.	Department Name	1 alphabet (A/B/C/D)
7.	Date of birth	2 digits (00 - 32), "-", 2 digits (00 - 13), "-", 2 digits (30 - 80)
8.	Age	2 digits (16 - 66)
9.	Salary	5 digits, ".", 2 digits (between 1000.00 and 50000.00)
10.	Overtime	4 digits, ".", 2 digits (between 0.00 and 2000.00)
11.	Perks	4 digits, ".", 2 digits (between 0.00 and 2000.00)
12.	Gross Salary	5 digits, ".", 2 digits (between 1000.00 and 52000.00)
13.	CPF Deduction	4 digits, ".", 2 digits (between 0.00 and 2000.00)
14.	Insurance Deduction	4 digits, ".", 2 digits (between 0.00 and 2000.00)
15.	Tax Deduction	4 digits, ".", 2 digits (between 0.00 and 2000.00)
16.	Total Deduction	4 digits, ".", 2 digits (between 0.00 and 2000.00)
17.	Net Salary	5 digits, ".", 2 digits (between 1000.00 and 50000.00)

Table 1: Specification of Attributes of Prototypical Payroll Data File

Serial No.	*Name of Attribute*	*Specification*
1.	Employee Number	2 digits, "-", 2 digits, "-", 3 digits
2.	Employee Name	maximum 30 alphabets with 1/2 white space
3.	Employee Address	maximum 45 alphabets with 2/3 white space
4.	Department Number	1/2 digits (1 - 20)
5.	Department Name	1/2 alphabets
6.	Salary	5 digits, ".", 2 digits (between 2000.00 and 30000.00)
7.	Allowances	4 digits, ".", 2 digits (between 0.00 and 2000.00)
8.	Gross Salary	5 digits, ".", 2 digits (between 2000.00 and 32000.00)
9.	Deductions	4 digits, ".", 2 digits (between 0.00 and 2000.00)
10.	Net Salary	5 digits, ".", 2 digits (between 2000.00 and 32000.00)

Table 2: Specifications of Attributes of Payroll Input Data File

Serial No.	*Name of Attribute*	*Specification*
1.	Student Number	2 digits, "-", 3 digits, "-", 3 digits
2.	Student Name	maximum 30 alphabets with 1/2 white space
3.	Student Address	maximum 45 alphabets with 2/3 white space
4.	Department Number	2 digits (10 - 20)
5.	Department Name	maximum 30 alphabets with 1 white space
6.	Date of birth	2 digits (01 - 31), "-", 2 digits (00 - 12), "-", 2 digits (60 - 80)
7.	Stage of Study	(B.S./M.S./Ph.D.)
8.	Class	1 digit (1/2/3)
9.	Grade	(A+/A/B+/B/C+/C/D+/D)
10.	Absolute Percentage	(90.0/80.0/70.0/60.0/50.0/40.0/30.0/20.0)
11.	Position in Class	1/2 digits (1 - 50)

Table 3: Specifications of Attributes of Student Grade Input Data File

Input file (Threshold)	**Unique** (0.12, 0.14, 0.17)	**Non-unique** (0.19, 0.08, 0.10)	**Similarity**
Payroll1	(0.02, 0.01, 0.01)	(0.14, 0.01, 0.04)	Yes
Payroll2	(0.02, 0.03, 0.02)	(0.12, 0.02, 0.06)	Yes
Grades1	(0.02, 0.01, 0.01)	(0.20, 1.10, 2.02)	No
Grades2	(0.02, 0.03, 0.02)	(0.22, 0.95, 1.64)	No

Table 4: Results of four input data files

the user must have prior knowledge of the subject. The manipulated data files should be properly designed so that the threshold distances are not too low or too high. If the threshold distances are too low then some data files that are actually similar are rejected as dissimilar. If they are too high then some data files that are actually dissimilar are wrongly selected as similar.

The threshold distances are more optimistic in nature as we consider the largest minimum distances as the thresholds. Some dissimilar data files which look like the prototypical data file may be wrongly declared as similar. This method is a result of an on-going research. Our ultimate aim is to match the attributes of the input data file with the attributes of the prototypical data file which will enable us to create the lost description file and completely explain the input data file.

References

[1] E. M. Awad and M. H. Gotterer, editors. *Database Management*, chapter 3. Boyd and Fraser, Danvers, Massachusetts, 1992.

[2] P. Drew, R. King, D. McLeod, M. Rusinkiewcz, and A. Silberschatz. Report of the workshop on semantic heterogeneity and interoperation in multi-database systems. *SIGMOD Record*, pages 47–56, September 1993.

[3] B. Everitt, editor. *Cluster Analysis.* Heinemann, London, 2nd ed. edition, 1980.

[4] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(3):77–85, 1990.

[5] James A. Larson, Shamkant B. Navathe, and Ramez Elmasri. A theory of attribute equivalence in database with application to schema integration. *Transactions om Software Engineering*, 15(4):449–463, April 1989.

[6] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th VLDB Conference*, pages 1–11, 1994.

[7] S. Navathe and Peter Buneman. Integrating user views in database design. *Computers*, 19(1):50–62, January 1986.

[8] Amit Sheth, James Larson, A. Cornelio, and S. B. Navathe. A tool for integrating conceptual

schemas and user views. In *Proceedings of the 4th International Conference on Data Engineering*, pages 176–183, Los Angeles, CA, February 1988.

[9] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationship. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[10] V. C. Storey and R. C. Goldstein. Creating user views in database design. *Transactions on Database Systems*, pages 305–338, September 1988.