Technical Report CS-99-02
10/18/99
Progress Report on Automated Data Cleansing

Jonathan I. Maletic
Andrian Marcus

The Department of Mathematical Sciences Division of Computer Science
The University of Memphis
Campus Box 526429
Memphis, TN 38152

jmaletic@memphis.edu, amarcus@memphis.edu

Introduction

This research is aimed at defining a framework for automated data cleansing. That is, given a large data set find and correct errors (semantic and syntactic) within the set. The underlying theoretical aspects of data quality research will be combined with existing (and possibly new) problem solving methods in software testing, data mining, knowledge based systems, and machine learning to address this framework. A basic understanding of the inherent problems faced by automated data cleansing will be uncovered and investigated.

This progress report describes the first steps adopted in the research, presents some initial results and findings, and suggests some future directions of the research program. The focus of the first phase of this research is concerned with defining and determining error types, and identifying error instances. Officer personal data from NPRDC is being utilized as a test bed for experimentation and validation of the proposed methods.

Problem Definition

The quality, correctness, consistency, completeness, and reliability of any large real world data set depend on a number of factors [Wang 95]. But the source of the data is often times the crucial factor. Data entry and acquisition is inherently prone to errors both simple and complex. Much effort is typically given to this front-end process, with respect to reduction in entry error, but the fact often remains that errors in a large data set are common. Unless an organization takes extreme measures in an effort to avoid data errors the field errors rates are typically around 5% [Redman98, Orr98]. Where the error rate is equal to the number of error fields over the number of total fields. There is a wide range of impacts to the organization including higher operational costs, poorer decision-

making, increased organizational mistrust, and diversion of management attention [Redman98].

The logical solution to this problem is to attempt to cleanse the data in some way. That is, explore the data set for possible problems and endeavor to correct the errors. Of course for any real world data set, doing this task "by hand" is completely out of the questions given the amount of person hours involved. Some organizations spend millions of dollars per year to detect data errors [Redman96]. A manual process of data cleansing is also laborious, time consuming, and prone to errors. The automation of the data cleansing process for large sets of data may be the only practical and cost effective way to achieve a reasonable quality level in a data set. While this may seem to be an obvious solution, little research has been directly aimed at this problem. Related research addresses the issues of data quality [Ballou89, Redman98, Redman96, Wang 95] and tools to assist in "by hand" data cleansing.

The Office of Naval Research (ONR) and in particular Navy Personnel Research and Development Center (NPRDC) have a very large interest in data quality. Large volumes of data are vital to support the day-to-day operations of the Navy and its personnel. Developing an underlying theory for the support of automated data cleansing will work to ensure the efficiency and validity of the decision-making processes that are required for this operation. For example, in personnel planning a large and diverse amount of data must be utilized in decision-making process to effectively assign duties and plan for changes. Unfortunately, as with any large data set, the decision making process can be subverted by erroneous entries and inconsistencies within the data set. Basic research and investigation of automated data cleansing methods addresses the inherent need of assessing and correcting copious data collections.

Research Approach

The process of automated data cleansing is multi-faceted and a number of problems must be addressed to solve any particular data cleansing problem. The following describes a generic framework for automated data cleansing:

- Define and determine errors types
- Search and identify error instances
- Correct the uncovered errors

In this initial phase, the research is focused on the first two aspects of the generic framework for automated data cleansing. The first part of the research includes the study of the domain of the data and building a rudimentary framework to define and determine errors.

The types of defined and identified possible errors are described in Appendix D. Identification of error instances thorough investigation of the existing literature on automated data cleansing and related topics (e.g., data quality, data mining).

The data used was provided by NPRDC.

C++ was chosen as implementation language. More than that a standard subset of the language was used in implementation to assure certain portability between platforms (e.g. Windows, Unix, DOS). A set of tools was built to search and identify the possible errors in the data set. The tools were tested on a reduced part of the original data set. Also, minimum domain knowledge was used. That allows to easily adapting the tools to a more general family of domains.

The following steps were done in this phase:

1. Analyze the data domain and the data set
2. Data reduction – extract the date type fields only
3. Implementation of a reusable data structure that allows manipulation of dates as integers
4. Compute statistics on data – tool implementation
5. Identify missing, wrong, and outlier values for some of the fields – tool implementation
6. Cluster the record space according to number and position of the empty fields – tool implementation
7. Identify patterns in the data according to the distribution of the field records per each field
8. Identify correlations between values of pairs of fields – tool implementation – ongoing

Each tool works independently of the others, but uses some of the results generated by the other tools. Some of the results provided by these tools are inconclusive (as described below). The other ones are intended to be included at the end in the data cleansing framework.

A data visualization tool was used to verify some of the hypothesis the research team made about the data, as well as the results of the tools.

Results

This paragraph presents the results of each step taken in the development of the research.

1. *Analyze the data domain and the data set*

   A snapshot of the data set for the "Officer personnel information system including midshipmen and officer candidates" and the associated data element dictionary was studied. A series of discussions with ONR personnel regarding the data set were conducted. The following conclusions were drawn:

- The only relevant information about the domain knowledge that can be obtained at this point is the domain range and format of the data elements. No constraints or correlations between attributes could be inferred at this point.
- Data mining techniques such as clustering, association rules identification, time series mining will be applied to gather more information on the data, regardless of the domain.
- In order to provide results applicable to a lager problem domain space, the research will not consider the domain ranges for the attributes as primary elements in the data cleansing process. It is considered that range checking for the data elements should be performed rather at the entry phase of the data.
- Considering this the research will focus on a more general approach, addressing the data cleansing issue from the perspective of a family of problem domains that are related to the ONR data set.
- The size of the data set is not too large (310 fields and approximately 50000 records). Exploitation of the data set and data cleansing are performed off-line. Therefore computation time is not a primary concern of the research project.
- Over 50% of the fields in the data set are empty.
- Over 50% of the data elements are date type, representing events.

2. *Data reduction – extract the date type fields only*
   Based on the previous results the research team decided to take into consideration only the date type fields from the entire data set. The data cleansing framework could be used in the future on any data set that contains date type data elements, regardless of the domain. Very little specific domain knowledge is included at this point in the framework.

   For the purposes of building and testing the tools in a time-efficient manner a subset of the entire data set was extracted and used. The 78 selected fields are mentioned in Appendix A. Each attribute is date type. In the entire data set there are approximately 110 date type fields. A set of 5000 records (roughly 10% of the entire data set) was used to test the tools. From now on the subset will be referred as the data set.

3. *Implementation of a reusable data structure that allows manipulation of dates as integers*
   A reusable data structure was created and implemented as a C++ class (*adate class*). The purpose of the data structure is to handle date type data in a standardized manner. It accepts as default date format the type of date used in the ONR data set. That is year-month-day. It can work also with other 5 date types and various date ranges. The date types are described in Appendix B.

   The main element of the data structures is that it converts dates to integers representing number of days passed between a reference date and the given date. The research team used this type of conversion to convert each date in the data set considering the ‘date of birth’ as the reference date. Thus each field was converted to an integer representing the number of days passed between the date in the field and

the date of birth, that is the first field. For those dates that did not have specified the month or the day we assigned 1 for each (January for the month and 1st for the day). The empty fields get a specific value, indicated by a constant (EMPTY). The fields with non-numeric values get a value specified by another constant (WRONG). Each field that contained the number 0 was considered empty. However, since this represents an inconsistency in the representation of the data it is mentioned as a possible error.

4. *Compute statistics on data – tool implementation*
   The first tool performs the following tasks: convert the date values into integers, output a new file containing these integer values (fields.txt), compute statistical values, output these values in a file (averages.txt), perform pair-wise comparisons between fields and output the result in a third file (comparisons.txt), assign to each value a string of bytes representing the empty (0) and non-empty (1) fields in the record and output these value in a forth file (hamming.txt). Each file is formatted in a tabular manner so that can be imported easily in any spreadsheet program or data visualizer. The structure of each file is presented in Appendix C.

5. *Identify missing, wrong, and outlier values for some of the fields – tool implementation*
   The second tool identifies the outlier, missing and wrong values in the data set. It uses the fields.txt and averages.txt files produces by the first tool. For each field it uses the average and the standard deviation, computed by the first tool. Based on Chebishev's theorem it identifies those records that have values in a given field outside a number of standard deviations from the mean. The number of standard deviations to be considered is customizable. The research team used 5 in the tests. The tool also identifies the 'wrong' values (non-numerical or negative). Among the 5000 records of the reduced data set, 164 contain outlier or wrong values. The tool in two cases points out missing values: if the first field (date of birth) is missing, or if only the first field is present and all the other ones in a record are empty. The results are written in the outliers.txt file that is described in Appendix C.

6. *Cluster the record space according to number and position of the empty fields – tool implementation*
   The first tool generates a set of strings of 0's and 1's associated with each record. Each string has as many elements as the number of fields. A 1 in the string represents a non-empty field on the same position in the record as the 1 in the string. A 0 in the string represents an empty field on the same position in the record as the 0 in the string. These strings are stored in the hamming.txt file.

   The team considered that the records could be treated as time series and thus existing methods in identifying patterns in time series might be applied on the data set. In order to do that the data set had to be portioned in subsets of records with the same number of non-empty fields on the same positions. We used the above-mentioned representation and clustered the records using the hamming distance. Initially we identified clusters having zero hamming distance between records. Unfortunately the

number of identified clusters was too high (4631 clusters for 5000 records). The largest cluster had only 98 records and the second largest only 29.

Therefore a new clustering method was applied such that the maximum hamming distance between the members of a cluster is customizable. We considered that records with approximately same number of empty fields (5% difference) would still contain significant patterns that could yield eventual outliers. This time the cluster number reduced significantly and the size of the larger clusters doubled. However the implemented clustering algorithm is too slow to be applied to the entire data set. Since the results are promising a new, faster algorithm will be implemented.

The results of these two clustering are contained in the files hcluster0.txt (for the 0 distance within the cluster) and hcluster.txt, respectively. The structure of the files is described in Appendix C.

7. *Identify patterns in the data according to the distribution of the records per each field – tool implementation*
   This tool is meant to identify patterns in records that apply to most of their fields. Initially the tool was supposed to use subsets of the data set containing the clusters with zero hamming distance. However, due to the small size of these clusters it was decided to apply the tool on the entire data set. Later we shall apply it on the clusters generated by the new clustering algorithm.

   The tool investigates the distribution of records according to each field and partitions the record set in subsets surrounding the mean (the number of subsets used was 6, one being used for empty fields). Once again this highlights some outliers. Each partition is ranked according to the number of records in it (i.e., partition number 1 has the largest size and so on).

   The team considered the following hypothesis: if there is a pattern that is applicable to most of the fields in the records, then a record following that pattern should have be part of the partition with the same rank for each field.

   The tool was applied on the entire data set and showed few records (0.3% of total number of records) that followed this pattern for more than 90% of the fields. The tool will be adapted and applied on cluster of records generated by the previous tool, rather than on the entire data set.

   The results of this tool are contained in the files orderedpartitions.txt and recappinpartitions.txt described in the Appendix C.

8. *Identify correlations between values of pairs of fields – tool implementation – ongoing*
   This tool that is under implementation and testing will identify correlations between pairs of records. It uses the results generated by the first tool, contained in the comparisons. txt file and described in Appendix C. The tool will analyze the

measurements contained in the file and based on the results will identify existing patters (e.g. the value of a given field is greater than the value another given field in 99.5% of the records). Based on these type of patterns new outliers will be identified.

Conclusions

The set of tools that has been developed and tested is a first element in the automated data cleansing framework. It addresses a rather general domain of problems, i.e. data sets that contain date type attributes. It is applicable to midsize data sets. In order to maintain its generality it makes little use of the domain knowledge. The only significant element extracted from the domain is that the first field in the data set (e.g. date of birth) should always have the lowest value in the record. This is however a condition that does not restrict too much the domain.

The set of tools successfully identified 5 types of potential errors in the data set (see Appendix D). Upon completion of the last two tools in the set eventually 2 more potential error types will be identified (e.g. outliers that do not fit the general pattern, and outliers that do not fit the pair-wise patterns). In addition the tool could be used to prove the existence or lack thereof of a general pattern involving most of the fields or just pair of fields.

As part of the automated data cleansing process, the set of tools is an essential part. Human interaction is still needed to prepare the date set and interpret the result data. However the set of tools as it is already performs better than a human expert even on midsize data sets.

The results generated by the set of tools will not only be used to identify the potential errors and attempt to correct them, but as a basis to define later some data quality metrics. The error correction issue that will be addressed later will differ form the current general approach since extensive specific domain knowledge will be necessary.

Future steps

Few more minor steps are needed to complete the first phase of the research (i.e. study the domain of the data and build a rudimentary framework to define and determine errors and identify the potentially erroneous instances).

1. Update the clustering tool with a faster and better algorithm.
2. Apply the tool to identify general patterns on the clusters rather than the entire data set and find the outliers if patterns are identified.
3. Identify the pair-wise patterns in the data set and find the outliers.

In order to continue the research to fulfill its stated goal, the next phase will contain the following steps.

1. Research the existing theories and practical results in the field that address the existence of patterns in data sets and how could that be used in defining certain data quality metrics.
2. Extend the framework with tools that based on the specific domain knowledge will attempt or suggest data correction.

References

[Ballou89] Ballou, D., Tayi, K., "Methodology for Allocating Resources for Data Quality Enhancement", *CACM*, Vol. 32, No. 3, pp. 320-329, 1989.

[Maletic95] *The Software Service Bay: A Knowledge Based Software Maintenance Methodology*, Ph.D. Thesis, Wayne State University, Detroit Michigan.

[Orr98] Orr, K., Data Quality and Systems Theory" *CACM*, Vol. 41, No. 2, Feb. 1998, pp. 66-71.

[Redman98] Redman, T., "The Impact of Poor Data Quality on the Typical Enterprise", *CACM*, Vol.41, No.2, Feb. 1998, pp. 79-82.

[Redman96] Redman, T., *Data Quality for the Information Age*, Artech House, 1996

[Wang95] Wang, R., Storey, V., Firth, C., "A Framework for Analysis of Data Quality Research", *IEEE Trans On Knowledge and Data Engineering*, Vol. 7, No. 4, Aug. 1995, pp. 623-639.

[Rus96] Rus, D., Gray, R., Kotz, D., "Autonomous and Adaptive Agents that Gather Information", in Proc. of AAAI-96 Workshop on Intelligent Adaptive Agents, 1996.

[Brodie80] Brodie, M., "Data Quality in Information Systems", *Information Management*, Vol. 3, 1980, pp. 245-258.

[Svanks84] Svanks, M., "Integrity Analysis: Methods for Automating Data Quality Assurance", *EDP Auditors Foundation*, Vol. 30, No. 10, 1984, pp. 595-605.

[Strong97] Strong, D., Lee, Y., Wang, R., "Data Quality in Context", *CACM*, Vol. 40, No. 5, May 1997, pp. 103-110.

[Wand96] Wand, Y., Wang, R., "Anchoring Data Quality Dimensions in Ontological Foundations", *CACM* Vol. 39, No. 11, Nov. 1996, pp. 86-95.

[Maletic98] Maletic, J., Moreno, M., "Identification of Test Cases During the Initial Phases of Software Development", to be submitted Feb. 1998.

[Guyon 96] Guyon, Matic, Vapnik, "Discovering Information Patterns and Data Cleaning", in Proc. KDD 1996, pp. 181-203

[Simoudis95] Simoudis, Livezey, Kerber, "Using Recon for Data Cleaning", in Proc. KDD 1995, pp. 282-287

Appendix A
Selected fields

No.	Filed name or indicator	No. in the original data set	Format – no. of digits
1.	Date of Birth	2	6
2.	Date of Initial Entry into Military Service	9	6
3.	Date of Initial Entry into Reserve Forces	10	6
4.	Pay Entry Base Date	11	6
5.	Pay Entry Base Date- Candidate 1	13	6
6.	Pay Entry Base Date- Candidate 2	14	6
7.	Pay Entry Base Date- Candidate 3	15	6
8.	Date of First Naval Commission	16	4
9.	Date of Active Commission	17	6
10.	Active Duty Base Date	18	6
11.	Date of Gain to Active Duty – Current	19	6
12.	Date of Gain to Active Duty-Initial	20	6
13.	ENSIGN	25	6
14.	LTJG	26	6
15.	LIEUT	27	6
16.	LCDR	28	6
17.	CDR	29	6
18.	CAPT	30	6
19.	COMO	31	6
20.	RDML	32	6
21.	Effective Date Current Pay Grade	35	6
22.	Date of Rank	36	6
23.	MINSERVR	38	4 proj. (*)
24.	Year Retire	39	2 proj.
25.	Estimated Loss Date	41	4, 6 proj.
26.	Transaction Loss Date	42	6
27.	Resignation - Date Received	43	4
28.	Date of Rank - Spot Promotion	47	4
29.	Projected Rotation Date	49	4 proj.
30.	BILSEQCD	52	6 proj.
31.	Accounting Category Code Date 1	58	6
32.	Accounting Category Code Date 2	60	6
33.	Accounting Category Code Date 3	62	6
34.	Accounting Category Code Date 4	64	6
35.	Accounting Category Code Date 5	66	6
36.	UNDCOM1	68	2
37.	UNDCOM2	71	2

38.	SCHCOMP1	74	2
39.	SCHCOMP2	77	4
40.	SCHCOMP3	80	4
41.	SCHCOMP4	83	4
42.	SCHCOMP5	86	4
43.	AOCPSP	90	4 proj.
44.	AOCPST	91	4
45.	Aviation Commission Date	93	4
46.	Aviation Service Entry date	94	4
47.	HVA	95	4
48.	Nuclear Service Control Date	96	6
49.	Additional Qualification Designation – Date 1	101	2
50.	Additional Qualification Designation – Date 2	103	2
51.	Additional Qualification Designation – Date 3	105	2
52.	Additional Qualification Designation – Date 4	107	2
53.	Additional Qualification Designation – Date 5	109	2
54.	Additional Qualification Designation – Date 6	111	2
55.	Additional Qualification Designation – Date 7	113	2
56.	Additional Qualification Designation – Date 8	115	2
57.	Additional Qualification Designation – Date 9	117	2
58.	Additional Qualification Designation – Date 10	119	2
59.	Additional Qualification Designation – Date 11	121	2
60.	Additional Qualification Designation – Date 12	123	2
61.	Designator Change Date 1	127	4
62.	Designator Change Date 2	130	4
63.	A1	136	4
64.	A2	137	4
65.	B1	140	4
66.	B2	141	4
67.	C1	144	4
68.	C2	145	4
69.	D1	148	4
70.	D2	149	4
71.	E1	152	4
72.	E2	153	4
73.	F1	156	4
74.	F2	157	4
75.	G1	160	4
76.	G2	161	4
77.	H1	164	4
78.	H2	165	4

(*) The fields marked with “proj.” contained projected dates, that is dates passed January 1st 2000

Appendix B
Date types

No	Type	Accepted format	Observations
0.	year, month, day	yymmdd, ymmdd, yymm, ymm, yy	Only 2-digit years
1.	month, day, year	mmddyy, mddyy, mmyy, myy, yy	Only 2-digit years
2.	day, month, year	ddmmyy, dmmyy, mmyy, myy, yy	Only 2-digit years
3.	year, month, day	yyyymmdd, yyyymm, yyyy	Only 4-digit years
4.	month, day, year	mmddyyyy, mddyyyy, mmyyyy, myyyy, yyyy	Only 4-digit years
5.	day, month, year	ddmmyyyy, dmmyyyy, mmyyyy, myyyy, yyyy	Only 4-digit years

Type 0 is the default date type handled by the adate class.

Two digit years are interpreted as follows:
- if yy<=CENTURY_21 then 21st century is considered
- if yy>CENTURY_21 then 20th century is considered

Appendix C
Structure of result files

File: **averages.txt**
Fields:
- Field number
- Minimum value
- Mean value
- Maximum value
- Standard deviation
- How many empty fields
- How many valid fields
- How many wrong fields – containing non-numerical values or negative numbers (representing dates earlier than the date of birth)

File: **fields.txt**
Fields:
- Record number
- Value for field no. 1
- Value for field no. 2
- …
- Value for field no. 78

These are the converted integer values.

File: **hamming.txt**
Fields:

- Record number
- Hamming value – the string of 0's and 1's representing the empty and non-empty fields respectively in each record

A 0 represent an empty field and a 1 represent a non-empty field.

File: **comparisons.txt**
Fields:

- Left-hand side field number
- Right-hand side field number
- How many comparisons were performed
- How many times the left-hand side value was greater than the right-hand side value
- Percentage of total number of comparisons
- How many times the left-hand side value was equal with the right-hand side value
- Percentage of total number of comparisons

File: **outliers.txt**
Fields:

- Record number
- Value for field no. 1
- Value for field no. 2
- …
- Value for field no. 78

The values are 0, 1, or 2. A 0 represents a good value, a 1 represents a wrong value (non-numerical or negative - representing dates earlier than the date of birth), and a 2 represents an outlier value. Only those record that contain outlier or wrong values are contained in the file.

File: **hcluster0.txt**
Fields:

- Record number 1
- Record number 2
- …

Each row represents a cluster. The number of the records in the cluster appears one after another in each row.

File: **hcluster.txt**
Fields:

- Record number 1
- Record number 2
- …

Each row represents a cluster. The number of the records in the cluster appears one after another in each row.

File: **orderedpartitions.txt**
Rows:

- Field number 1:
  Partition 1: number of records
  Partition 2: number of records
  …
  Partition 6: number of records
- Field number 2
  Partition 1: number of records
  Partition 2: number of records
  …
  Partition 6: number of records
- …
- Field number 78
  Partition 1: number of records
  Partition 2: number of records
  …
  Partition 6: number of records

Partition number one within each field has always the largest size and so on.

File: **recappinpartitions.txt**
Fields:

- Record number
- Number of fields in which the record appears in partition number 1
- …
- Number of fields in which the record appears in partition number 6
- Percentage between the maximum number of fields from the partitions and the total number of fields

Each row represents a record.

Appendix D
Potential error types

So far the implemented tools identified the following potential errors.

1. Non-numerical value in a date type field – highlighted in outliers.txt and fields.txt
2. A value that represents a date earlier than the date of birth – highlighted in outliers.txt and fields.txt
3. Missing value in the data of birth field – highlighted in fields.txt
4. Too many empty fields – every field but one are empty in a record
5. Outlier values in a record – highlighted in outliers.txt