British Journal of Mathematics & Computer Science
19(4): 1-17, 2016; Article no.BJMCS.28861
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

iCleaner: A Data Cleansing Tool for Outlier Detection in a Data Warehousing Environment

Kofi Sarpong Adu-Manu[1*], John Kingsley Arthur[1], Joseph Kobina Panford[2] and Joseph George Davis[2]

[1] *Department of Computer Science/Information Technology, Valley View University, Accra, Ghana.*

[2] *Department of Computer Science, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana.*

Authors' contributions

This work was carried out in collaboration between all authors. Author KSAM proposed the concerned problem, designed the study, implemented the algorithm, and wrote the first draft of the manuscript. Authors JKP and JGD reviewed the algorithm, managed the analyses of the study and evaluated the experimental results. Author JKA managed literature searches. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/28861
Editor(s):
(1) Mohamed Rabea Eid Said, Department of Science and Mathematics, Assiut University, Egypt.
Reviewers:
(1) Chi (Harold) Liu, Beijing Institute of Technology, China.
(2) Joseph Elias Mbowe, The Nelson Mandela African Institution of Science and Technology, Tanzania.
(3) Marwa Salah Farhan, Helwan University, Egypt.
Complete Peer review History: http://www.sciencedomain.org/review-history/16796

Original Research Article

Received: 9th August 2016
Accepted: 11th October 2016
Published: 4th November 2016

Abstract

The implementation of Data Cleansing (DC) in Data Warehousing (DW) is essential in recent years. Organizations around the world generate huge amount of data from their day-to-day activities for their operations. These organizations will not survive if the data they generate remains *dirty or erroneous*. There are errors or outliers that make the data become *dirty* such as data entry errors, outdated data in the database, data migrated from old databases, and changes made at the source repository. The changing needs by customers to update their records

**Corresponding author: E-mail: kadumanu@ur.rochester.edu;*

(for example customer attributes such as marital status, phone number or address changes with time) cause records to become obsolete and reducing the quality of data. In order to obtain high quality data over time, the data requires cleansing. The proposed Integrated Cleaning (*iCleaner*) tool is developed to facilitate the data cleaning process and to address the problems associated with duplicated records. In addition, the proposed cleaning tool is able to detect and update missing data by merging key columns within the records. The system is flexible to use and comes with a convenient user-friendly interface designed for the data cleansing process. We provide an efficient, but simple algorithms designed to perform these functionalities and provide the running time for the system performance.

Keywords: Data cleansing; data quality; database system management, data warehousing; automatic periodic scheduler; duplicates; missing data; data integration; data quality tools.

2016 Computer Science Subject Classification: 53C25, 83C05, 57N16.

1 Introduction

The advances in technological development have seen rise in businesses generating huge data from daily transactions around the globe. The recent adaptation of new tools to preserve the integrity of these huge amount of data by businesses have seen an increase in the realization of their strategic goals [1] and maintained higher credibility with their customers and suppliers [2]. Data warehouse (DW) is a modern proven technique for handling and managing diversity in data sources, data format, and structure. Recent advances in database technologies are leading to the proliferation of different kinds of information design with independent supporting hardware and software. This technology is developed to integrate heterogeneous information sources for data analysis purposes [3]. DW integrates data from several operational sources (databases) all over the organizations [4].

In recent times, data collection have become ubiquitous, large organizations collect data not only for record keeping, but to support variety of data analysis tasks that are vital in the achievement of the organization's mission and exponentially support the growth of the organization [5]. As a result, the integrity of the data should be of much interest to stakeholders. When the data gathered is found to contain outliers, data cleansing process is employed to deal with the outliers. Researching into various aspects of data cleansing (DC) to find procedures to automatically or semi-automatically identify and correct outliers (errors) in large datasets then become relevant. DC typically deals with the identification of corrupted data in large datasets in order to enhance its quality causing the data to be *dirty* [6]. Data Cleansing is an activity performed to detect and correct the data quality problems (i.e., caused by incorrect, missing, and duplicated data), and inconsistencies from dirty data [7][8]. It is vital for organizations to manage data by constantly monitoring and cleaning the datasets obtained from their databases and file systems to maintain the quality of data. Data cleansing therefore an important step in the integration of organizational data [9]. According to the Data Warehousing Institute (DWI), data quality problems cost businesses in United States of America over 600 billion dollars per year [7]. Data quality problems typically occur in one or more values of a single attribute and multi-attribute quality problems [10][11]. Hence, developing an efficient data cleansing tool to detect and remove outliers such as single or multiple attributes automatically from organizations' database systems will support and increase the quality of data to enhance business solutions [12]. These outliers in datasets (records) result in inconsistencies in data analyses and reports which affects the performance of the organization in decision making. In our previous work, we investigated some existing commercial data cleansing tools taking into account their strengths and weaknesses and we also proposed a conceptual framework which included an automatic periodic scheduler (APS) for the DC process [7][13].

In this paper, we employ system analysis and design methodology tools to develop and test the performance of our proposed system *(iCleaner)*. We obtained data from students database as input to test the performance of the system. Several datasets of varied sizes (ranging from Kilobytes to Gigabytes) were used for the system testing. The research results strongly revealed that maintenance of the clean data within periodic intervals, the file format, the error type, the number of processors and memories are major factors in the data cleansing process. Two main strategies are proposed: 1) the strategy of storing all the strings resulting from the *addition of columns* in a dictionary to identify duplicates and 2) the strategy of *comparing key columns* to identify empty or null fields. These strategies made it easy for searching through the datasets.

The rest of the paper is organized as follows. In Section 2, we provide the motivation of the paper. The background and related literature are presented in Section 3. The conceptual model of the proposed algorithms are discussed in Section 4. The implementation and testing are discussed in Section 5 and we conclude in Section 6.

2 Motivation

The concept of data warehousing and data cleansing have gained popularity in the business and in the academic community. Companies are challenged with the increasing number of data collected from its customers and employees. Data transformed into information are relevant for decision making by management for reporting and the purposes of data analysis. The task of keeping large datasets clean (free from errors), consistent, and data integration turns to be a difficult for most organizations [14][15][16]. Moreover, identifying and fixing data errors often require manually inspecting data, which can quickly become costly and time-consuming. Ignoring the effects of *dirty data* on the organization is potentially dangerous. Analysts have to choose between facing the cost of data cleaning or coping with results of unknown inaccuracy [17]. To ensure that all managerial activities, functions, and decisions are smoothly undertaken, there is a compulsive requirement for the data to be meaningful. Data collected from different sections or units in organization representing any such entity as name, date employed, height, weight etc are collected and processed for management to keep track of customers, and employees, and track the volume of data from its numerous branches, analyze and interpret the data to support decision making processes, and support the business process.

During data collection, several errors occur. Among these are omissions, data entry problems and double entry problems rendering databases redundant. The data residing in operational databases, though of sufficient quality for transactional procession, contain duplications, inconsistencies, errors, missing data, and other related issues that make the data unsuitable for decision support data warehousing, without considerable cleansing of the data [18][16][19]. The researchers are motivated to undertake this research in order to develop a data cleansing tool to support companies, government agencies, educational institutions, etc., that collect and process data for analysis to get rid of common errors. This paper is aimed at implementing an automatic correction method (data cleansing) to detect and remove duplicates and incomplete data in different parts of organizations database.

3 Background and Related Literature

Organizations are faced with the data quality problems and they spend huge amount of money on poor data annually. It is expensive to deal with poor data especially when reporting on the company's performance over a period of time. Poor data also affects business decisions and customer relationship management [20]. Data tends to be the core business asset that needs to be managed

with optimal quality if an organization is to generate a return from it. Researchers have attempted to solve the problems related to poor data especially when the data are to be stored in a data warehouse. In data warehousing, data cleaning becomes relevant when data is drawn from several databases to be merged at one point. There are a number of ways to represent or refer to the same entity in different data formats. Erroneous data in databases can be reported and captured in a data warehouse.

There are many issues in DC that researchers have tackled using different tools and approaches. Of particular interest is the search context for what is called in literature and the business world as *dirty data*. *Dirty data* are in the form of missing, duplicate, incorrect, or inconsistent data values [17][21][22][23]. Recently, [24] proposed a taxonomy for *dirty data*. The search context is an important issue which have attracted the attention of researchers and practitioners. It is the first step in defining and understanding the data cleansing process. *Dirty* data is caused as a result of misuse of abbreviations, data entry problems, control information hiding, missing fields, spellings, outdated code [18][16]. Data cleaning addresses the issues of detecting and removing errors and inconsistencies from data to improve its quality [25]. In general, the architecture for DC consist of five different stages: 1) Data Analysis to detect the inconsistencies; 2) Definition and choice of transformations; 3) Verification of the correctness and effectiveness of transformations; 4) Execution of the transformation; and 5) Feedback of cleaned data [26], [27]. Following these stages in data cleaning process, business owners are able to audit the data to find discrepancies, choose transformations to fix the problems, and apply transformations on the dataset to obtain a clean data [28]. To achieve this, auditing and transformation tools are required. Since the data have many hard-to-find special cases, this process of auditing and transformation must be repeated until the *data quality* is good enough. This approach has two problems − lack of interactivity and requires much user efforts [28].

In the early 2000's, different commercial data quality tools were proposed to solve data quality problems by [29][16][12][23]. But most of these data cleansing tools basically audits single specified fields, although a number of them tackle multiple fields [7]. In recent times, data quality tools such as Talend Open Studio, DataCleaner, WinPure, Data Preparator, Data Match, DataMartist, Pentaho Kettle, SQL Power Architect, SQL Power DQguru, ActiveClean, Sampleclean, and DQAnalyzer have been developed to integrate functions such as data profiling, data integration, and data cleaning. These tools are able to audit multiple fields and are tailored to specialized use cases (i.e., individual aggregation queries and convex models, etc.) [30][17][31]. Apart from these data cleansing tools, there are other approaches that are employed to address the data cleansing problem. In one such approach, the authors determined the common cause of errors for constraint-based data cleansing [9]. In another approach, a system for big data cleansing was designed to tackle efficiency, scalability, and ease-of-use in data cleansing [32], and an automatic data cleansing using longest common subsequence (LCS) algorithm was proposed to find the longest subsequence common to all sequences in a set of sequences for error correction to find incorrect city names in datasets [33]. Finally, in a recent survey conducted by [8], three important themes were evaluated among data analysts and infrastructure engineers from industry and academia. These included the iterative nature of data cleaning, the lack of rigor in evaluating the correctness of data cleaning, and the disconnect between the analysts who query the data and the infrastructure engineers who design the cleaning pipelines. The authors noted that despite the amount of work done by the data cleaning community in developing interactive data cleaning tools and approaches, there still remain opportunities for systems that facilitate and automate rapid human-in-the-loop interactivity, hence our proposed framework.

3.1 High quality data

Data can be described as high quality when it satisfies a set of criteria. In general, quality is defined as an aggregated value over a set of quality criteria [34]. High quality of data warehouse is a key to make smart strategic decisions. Hence, correctness of data is essential for well-informed and reliable decision making [29]. The data cleaning is a program that performs to deal with the quality problems of data extracted from operational sources before their loading into data warehouse [27]. Data quality is data that is fit for use by data consumers [10][35]. Data cleaning addresses the issues of detecting and removing errors and inconsistencies from data in order to improve the quality of the data. Data quality has many attributes [36]. An aggregated value over the criteria of integrity, consistency and density is referred to as accuracy. Intuitively this describes an exact, uniform and complete representation of data collection not containing any defined anomalies except for duplicates [4][18].

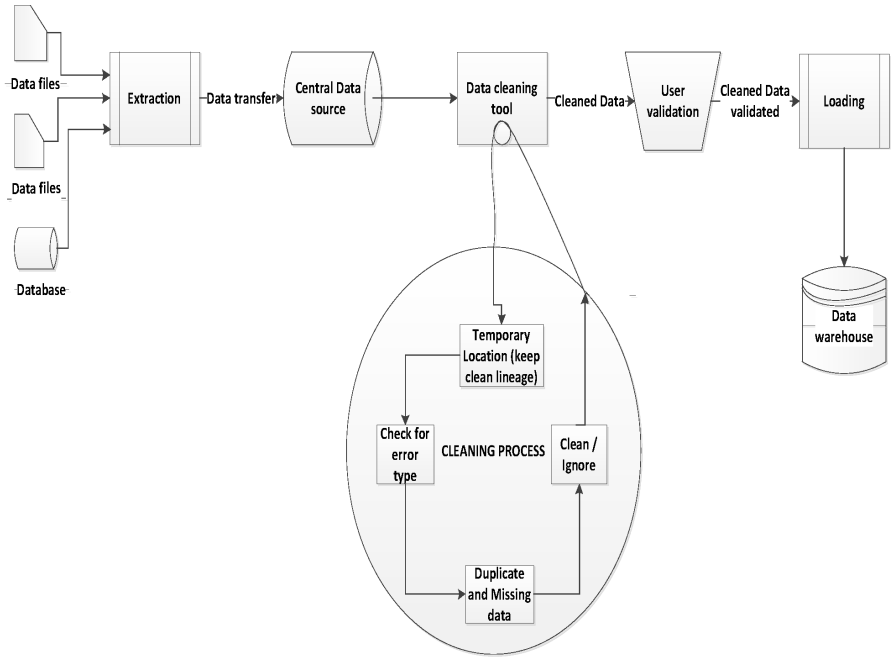

Fig. 1. Data cleansing framework

3.2 Data cleaning work flow

Data cleansing work flow follows a predefined process from the beginning to completion to achieve a specific task [37]. The data cleaning process is complex and begins when the data is imported into the application, for example, AJAX [29], IntelliClean [16], Potter's Wheel [12], and ARKTOS [23]. In some cases, the data cleaning tools listed above are unable to remove completely all the anomalies and therefore the user involvement in the data cleaning process cannot be overlooked. The user has the choice of selecting the type of error within the database to perform the cleaning and verify that (s)he is satisfied with the cleaning process. The user then pushes the clean data, void of duplicates and/or missing field items to the centralized location to complete the verification process.

In DC, statistical methods are used for auditing the data to detect some form of data anomalies and contradictions [1][38]. The process of auditing the data supports the achievement of high quality data as the end product [29][1] which is useful, relevant, accurate, and concise. A proper workflow can be used as an alternative solution for common errors such as typographical errors [38] which can be corrected, for example by considering the layout of the keyboard. Hence, whenever the data cleaning process or workflow fails to correct errors automatically, then errors are corrected by the user manually [1].

4 Data Cleansing Design

The conceptual framework in [13], illustrates how data is extracted, cleaned, transformed, and loaded from source computers into the central database. The design of our proposed system takes into consideration the file format, the user interactivity, the cleaning process, and maintenance. Another architecture is presented in [33], it consists of the user interface, longest common sequence (LCS) algorithm, file selection for input, and a database. In what follows, we provide detailed discussion of the data cleansing framework implemented in this work.

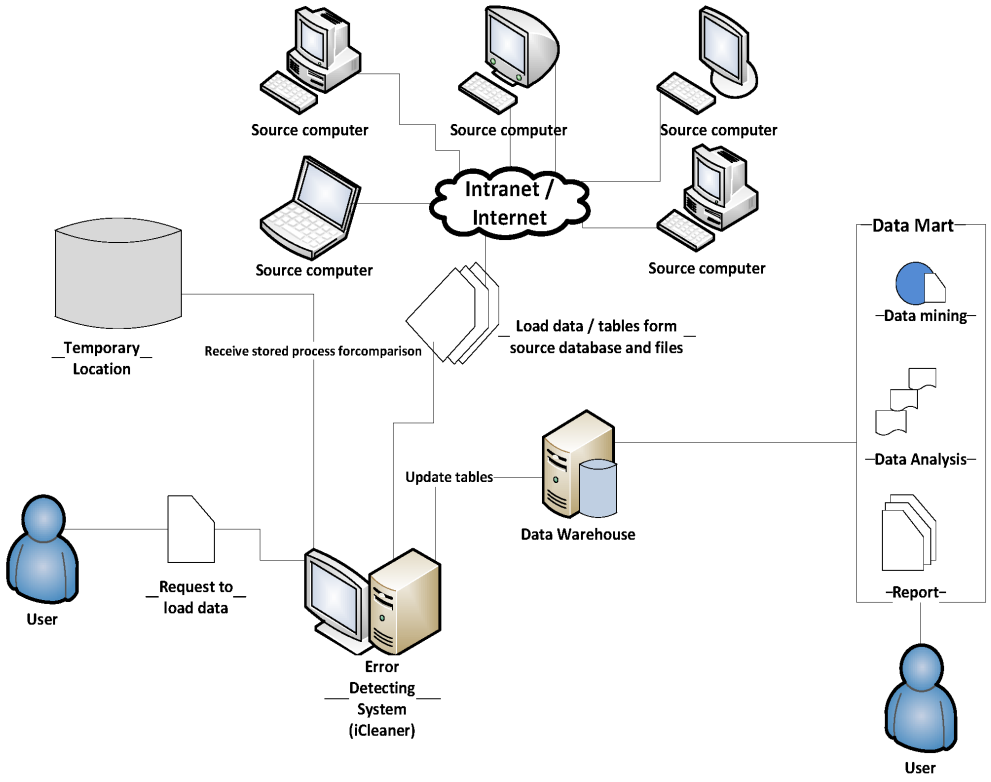

Fig. 2. Data cleansing system model

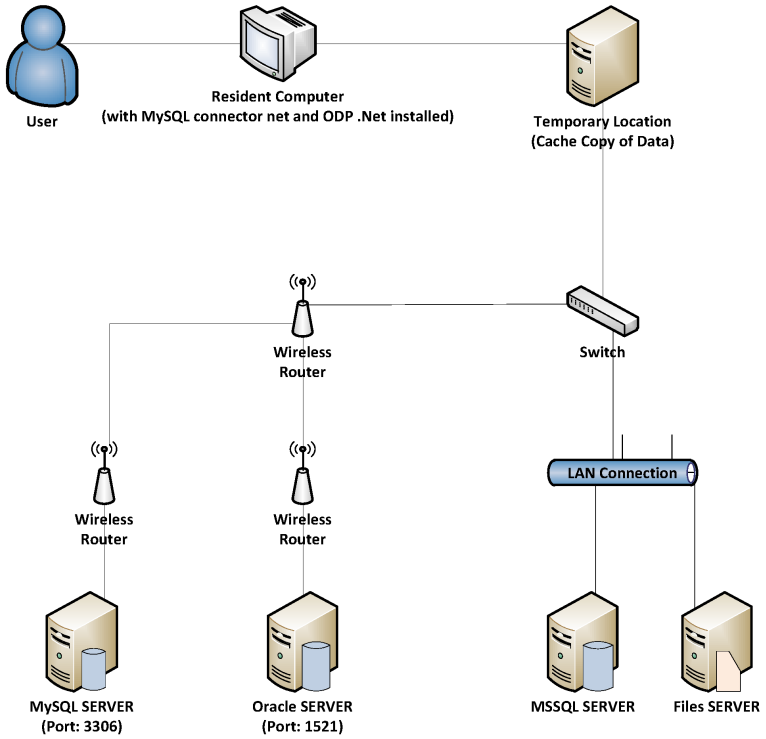

Fig. 3. Data cleansing architecture

4.1 Data cleansing framework

The data cleansing process usually consist of three main steps to achieve a clean data - detecting the errors, selecting and applying the most appropriate methods to correct the errors, and preventing the errors from happening again [39]. In this section, we present a framework for the data cleansing tool that follows the steps provided above. We provide a flowchart that illustrates the process carried out in the DC process. The data cleansing framework shown in Fig. 1, illustrates how data is extracted, cleansed, transformed, and loaded from the source computers into a central database. [10] described the most popular methodologies in data quality management, namely Total Information Quality Management (TIQM) and Total Data Quality Management (TDQM) required for data quality management activities. Data cleansing activities forms part of the TIQM activities which seeks to improve data quality by identifying the data sources that require data cleansing, extracting and analyzing the relevant source data for anomalies and patterns, standardizing the data, i.e. the reduction of synonymously used data values and patterns, performing manual or automated correction of the data, consolidating duplicated data, analyzing data defect types, transforming the data to target state, data aggregation and audit control(data warehouse-specific). The conceptual framework in Fig. 1 is designed to follow the processes in TIQM. The framework is divided into three phases: a) the data extraction and transfer phase, b) the data cleansing phase, c) data validation phase. In the first phase, the user extracts data (i.e., text files and tables from databases) from different operational sources and transfers these data into a central repository (i.e., a database). The goal is to store all the different data formats from the different text files and databases into a centralized location for easy access and cleaning. The data data cleansing phase is an iterative process tailored to the requirements of a specific analysis task [30]. At this phase, the data cleaning

tool identifies the appropriate data formats, analyze for anomalies, and performs standardization. This process continues until *clean data* is obtained. When the data is loaded into the data cleansing tool *iCleaner*, the following steps are performed: 1) A cache copy of the data is created temporarily in the system. 2) The type of error to remove is selected by the user (e.g. duplicate or missing data). 3) When *duplicate* is selected, then the *iCleaner* system will scan through to detect all duplicates by adding all the columns in each rows to obtain a unique value referred to as the *key*. 4) When the error type selected is *missing data*, then the rows are updated accordingly by comparing rows based on key columns. Finally, the *clean data* is validated by the user before uploading the data into the desired location on a secondary storage device or into the data warehouse, otherwise the data is sent back for further cleansing. This whole process is automated. The details of identifying each of the error instances are presented in the sections that follow. In system implementation, we illustrate the data cleaning process using the system model in Fig. 2. The system model shows the operation of the error detecting system (iCleaner). The system is connected to various source computers via the Internet or Intranet. Upon request, the user loads data into the temporary location (database) to begin the cleansing process. The clean data is updated into the appropriate tables in the DW. This process is further illustrated with Fig. 3, where different database servers (e.g., MySQL, MSSQL, and Oracle) and a file server sends offloads the requested data to cache server (temporary location) for cleaning and user action.

4.2 Detecting duplicates - addition of columns

This section describes the *addition of column* method used for the identification of duplicated data. To achieve this, *the individual columns in each row are added to obtain a single value*. The system is designed to search through the datasets using the binary search method. This search method is used in order to obtain optimal performance and response time. In order to detect duplicates in any data set, all the key columns are concatenated (or added) together and converted into a single string. It is imperative to note that there is a consistency among columns within the various tables in a database. In the method of detecting duplicates, each row that has been concatenated or added generates a single value (this is a string). From Table 1, concatenating the fourth row (i.e. adding each value in the column; here each value represents any data type found within the column), the following string is generated *2125642MrArthurJohnKingsleyMale3/14/1993Mankesim*. From Table 1, after the string has been generated, each of these strings are stored in a single value dictionary and they are compared with the rest of the generated strings in the dictionary to detect all occurrences of the same string (i.e. duplicates). Each value item is used in the comparison process. During the comparison of the strings, each single value or string that does not recur in the list is considered as unique and it is stored in a hash table (unique key dictionary). A unique key is defined as the single value or string that has no recurrent (repeated) value in the single key dictionary. When the first string is compared to the other strings and a match is found then that string value is considered duplicated and this string value is stored in non-unique key dictionary. It should be noted that every key in the dictionary acts as a pointer to the record within the rows fetched from the main table. To identify duplicates, we define the error instance as E, the dataset is represented as D, each row instance is denoted by R, where R=$\{d_1, d_2, d_3,..., d_i,..., d_n\}$ and where d_i is the i^{th} data value. Also, each record is denoted as F, where F=$\{f_1, f_2, f_3,..., f_i,..., f_n\}$ and where f_i is the i^{th} field and f_1 is a subset D. Each table is represented by T with r records, where the value of d_i is also a subset of T. We also define C to represent the summation of columns, where C represents duplicated or unique values, S_d denotes all duplicated rows and S_u all unique rows representing the variables to store differences between two columns C_k and C_{k+1}. We compute S_d by assuming that k=*0* and i=*0*, then $S_d = C_k - C_{k+1} = 0$, this results in the detection of duplicated records. Hence, we calculate duplicate as, $dup = \sum_{k=1}^{n} C_k - C_{k+1} = 0$, whereas unique values are determined as follows: $S_u = C_k - C_{k+1} < 0$ and $uq = \sum_{k=1}^{n} C_k - C_{k+1} < 0$ to obtain unique keys. On the other hand, to determine duplicated rows we concatenate all the columns within a particular row. The

string value obtained should not be equal to the addition/concatenated value of any other key value within table with n number of rows. If this occur then that record is a duplicated record. Assuming $k = 0$, $key =$"", $m = 0$, and $i = 0$, where key is the unique identifier, m is the variable holding the concatenated values of all fields in a specified row, i is the value of each field and R_n is the last row within the dataset. We determine the duplicated data as shown in Table 2.

Table 1. Sample data of students records

Student ID	Title	Last Name	First Name	Other Name	Gender	DOB	POB
2125600	Miss	Adu	Christiana	NULL	Female	6/30/1981	Tema
2125620	Miss	Agbeko	Mavis	NULL	Female	NULL	Kumasi
2125631	Mrs	Afrifa	Yvette	Akosua	Female	10/25/1987	Accra
2125642	Mr	Arthur	John	Kingsley	Male	3/14/1993	Mankesim
2125633	Mr	Ofori	Amanfo	Emmaneul	Male	1/12/1991	Sefwi Wiaso
2125624	Miss	Aidoo	Patience	NULL	Female	5/15/1978	Tarkwa
2125627	Miss	Akafia	Lawrencia	NULL	Female	6/4/1980	Accra
2125677	Mrs	Okoe	Theodora	NULL	Female	7/14/1984	Sandema
2125648	Mr	Ampofo	David	NULL	Male	NULL	Navrongo
2125611	Mr	Poku	Kwame	Nana	Male	11/13/1984	Tamale
2125610	Miss	Opoku	Berlinda		Female	9/23/1991	Wenchi
2125688	Mr	Danso	NULL	Prosper	Male	12/10/1982	Koforidua

Table 2. Searching for duplicates

Algorithm 1

while (k = R_n)
{
$m = R_k \times F_n + R_k \times F_{i+1} + ... + R_k \times F_n$
$m = \sum_{k=0}^{n} \times \sum_{i=1}^{n} R_k \times F_{i-1}$
if (key.contains (m))
{
println("Yes, duplicate found, clean error instance E")
}
else
{
println("No, duplicate not found, ignore error instance E")
key = key + m
}
k = k+1
}

4.3 Detecting missing data−key columns

According to [39], data can be missing for a number of reasons. For example, a client may have declined to answer a question, a data inputter may not have had the data to enter into the database,

and so on. Useful data would be wasted if we ignored all rows with a missing value. Hence, to detect missing data, this system specifically identifies fields with *Null values and Blanks*. This method used to detect missing data (that is, an empty field) is based on the key columns within the rows. The key columns in this case are determined by the user. Key column is explained as a unique field in a record as shown in Fig. 4, the system/tool is designed to allow the user to have privileges in selecting and determining the key columns to use as the search criteria. In searching for missing data, we propose a technique where in each row the key columns are verified for *blanks or NULL*. In this case, two pointers are distinctively relevant− the blank or null column pointers. To this point, the missing data is identified in a row when one of the column data is not available or present. The algorithm identifies particular columns of interest referred to as *key columns*. For example, in Table 1, the column with heading *OtherName* is not considered a key column because generally, not all persons have other names but considering columns such as *FirstName, LastName, DOB and POB*, such fields are attributes required about every student. This is further illustrated in Fig. 4.

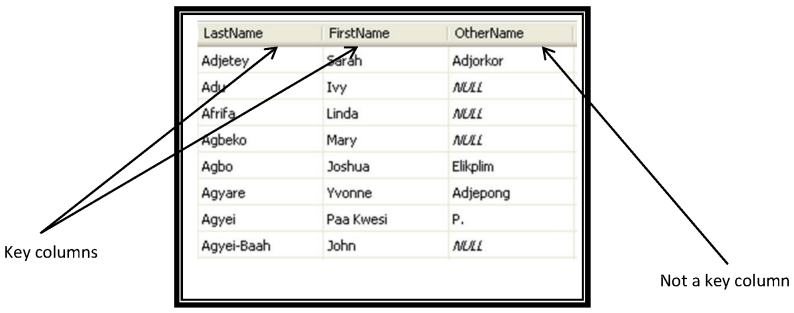

Fig. 4. Key column identification

Table 3. Searching for missing data

Algorithm 2

for a = 0 to D
{
for i = 1 to n
{
if $(d_a \times F_i)$ = NULL
{
return true
println(Missing data found in the row)
}
else
{
println(No Missing data found in the row)
}
}
}

In order to detect any missing fields in a particular data set, the following procedure/method was experimented. A key column in a row that requires verification is selected by the user for *null values or blank spaces*. The system then performs a search in the entire database for the selected key columns. These key columns are compared to all the fields in the database. When a related field is found in the comparison operation of the key columns with the other columns, the system automatically updates the field with the new data value and merge the two fields. On the other hand, if no such record is found the system maintains the null value or empty fields and imports the row to a new table for verification by the user before pushing it into the DW. Finally, when the system searches the database and the key columns selected is not equal to null, then there exist no empty field. We denote d_a and F_i as row and field respectively. We determine the missing data as shown in Table 3.

4.4 System complexity

The process of detecting and eliminating approximated duplicated records and missing data from operational database into a Data Warehouse (DW) and applying the Data Cleansing (DC) technique is a very complicated task. In order to identify and remove duplicated records and missing data from different sources, the column addition technique using binary search tree and pair-wise strategy for comparing various columns is proposed. The proposed system is expected to generate records from the appropriate tables in the databases 100% of the time. The speed of the proposed system will depend on the hardware requirement rather than the systems characteristics. The proposed algorithm exhibited a quadratic time or order n^2. Hence, the running time of the algorithm is $\mathcal{O}(n^2)$. This is so because the running time of the algorithm depends much on the input size of the dataset that is loaded into the system (*iCleaner*) to perform the cleansing process. The algorithm's running time does not depends data loaded into the cache copy but during the cleansing process.

The proposed system can run with the minimum hardware requirement but comparatively performs better on hardware with good specifications. The algorithm's space complexity is essentially related to the memory size. After running the system several times with different input sizes, we realized that the system does not uses so much of memory to process data because the data is loaded into cache. When the input size is very huge it still uses a reasonable amount of memory and processing time for the cleansing operation to complete.

5 Implementation

The cleaning tool presented in this paper will benefit organizations, institutions, and companies in the following ways: 1) increase the accuracy and consistency of records; 2) facilitate proper data analysis and 3) increase the integrity of the organization's data for reporting and strategic planning. The proposed data cleansing tool is easy to use, saves a great of time, and allows data to be imported and exported within the shortest possible time. In order to develop the tool, a series of steps were followed to design the system. The flowchart in Fig. 5 depicts the flow of operations in the data cleansing model. The flowcharts begins by accepting the type of error and ends with the user validation and then terminates. The system flowchart serves as the roadmap for the data cleansing tool because it is showing how all the major components interact and fit together.

5.1 Deployment

The implementation of the integrated algorithm for data cleansing is achieved through the coding of the pseudo-code algorithm inferred from Fig. 5. The proposed system was developed using C-sharp programming language. The Integrated Cleaner (iCleaner) is a powerful data cleansing tool developed to clean and correct duplicates and missing data. The proposed system is capable

of cleaning data retrieved from Database and File Servers such as MSSQL, MySQL, Oracle, and Text files with these extensions: .txt and .csv, and Excel files with the following extensions: .xls, .xlsx, xlsm, and xlsb as illustrated in Fig. 6. Once you obtain the necessary credentials (i.e., server name/instance, user name, and password) for any of the relational databases (i.e., MSSQL, MySQL, and Oracle), the user will be able to access all the tables within the selected database. The data is imported into the *iCleaner* for cleansing. The clean data is exported to the Data Warehousing system.

5.2 Graphical User Interface (GUI)

User interface design requires easy usage of all the component parts of the system and should allow for effective use of the system. The user interface is intuitive as compared to other tools in the market. The GUI shows the various procedures the user must implement to clean data after loading the data from different operational sources. Users begin by selecting the database or text file, enter the required server name, username, and password in order to load the data to begin the data cleansing process. For example, in Fig. 6 a user can enter the server address *192.128.15.10* at the *data source*, user name *schoolMSystem*, and *passw0rd* at the password field. The user selects the appropriate table from the drop-down list at the *initial catalog to use* and then clicks on the *Test Connection* to load a cache copy of the data into the *temporary location* as illustrated in Fig. 7. The data operation (i.e., removing duplicates or missing fields) is selected by the user and the cleansing process is initiated from this point.

5.3 System testing and performance analysis

We tested *iCleaner* using real datasets from a student database. The database had over 4000 records and a sample table is shown in Table 1. The results are discussed in the subsections that follow.

5.3.1 Duplicates

In order for the tool developed to meet its expected performance for detecting and cleansing duplicates, the cleansing tool will take advantage of the advances in processor and memory development in recent times and in the near future. The system was tested with a minimum data size of 5GB and a maximum data size of 30 GB. It was tested on a computer system with a single processor, and multiple processors both with a 64 MB of memory. It is platform independent. The performance of the system greatly improve on devices with multiple processing power as shown in Fig. 8. This characteristic is known as the scale-up factor, i.e. the amount of speed and the throughput gained by the application depended on the systems performance, which is a key factor.

5.3.2 Missing data

With respect to missing data set, the data cleaning or cleansing tool relies on the *key columns* selected by the user. These columns are used as the search criteria during the data cleansing process. At this stage, all the columns in the table can be used as part of the search criteria or single field or multiple fields can be used as the criteria for searching for records with missing fields. There is tradeoff between processing time and memory usage when all fields are used as the search criteria.

Therefore, the best policy will be to use the maximum number of relevant fields to accomplish the desired results. Fig. 9 shows that the cleansing or cleaning time achieved in identifying missing data is far less than the time for cleaning duplicates. When performing data cleaning or cleansing to

remove duplicates, all the columns are factored into the searching criteria. In the case of identifying missing data fields there is an option for the user to make his choice in discovering missing records. Fig. 9 shows the running time for the data cleansing tool/system in identifying missing follows quadratic function which is $\mathcal{O}(n^2)$.

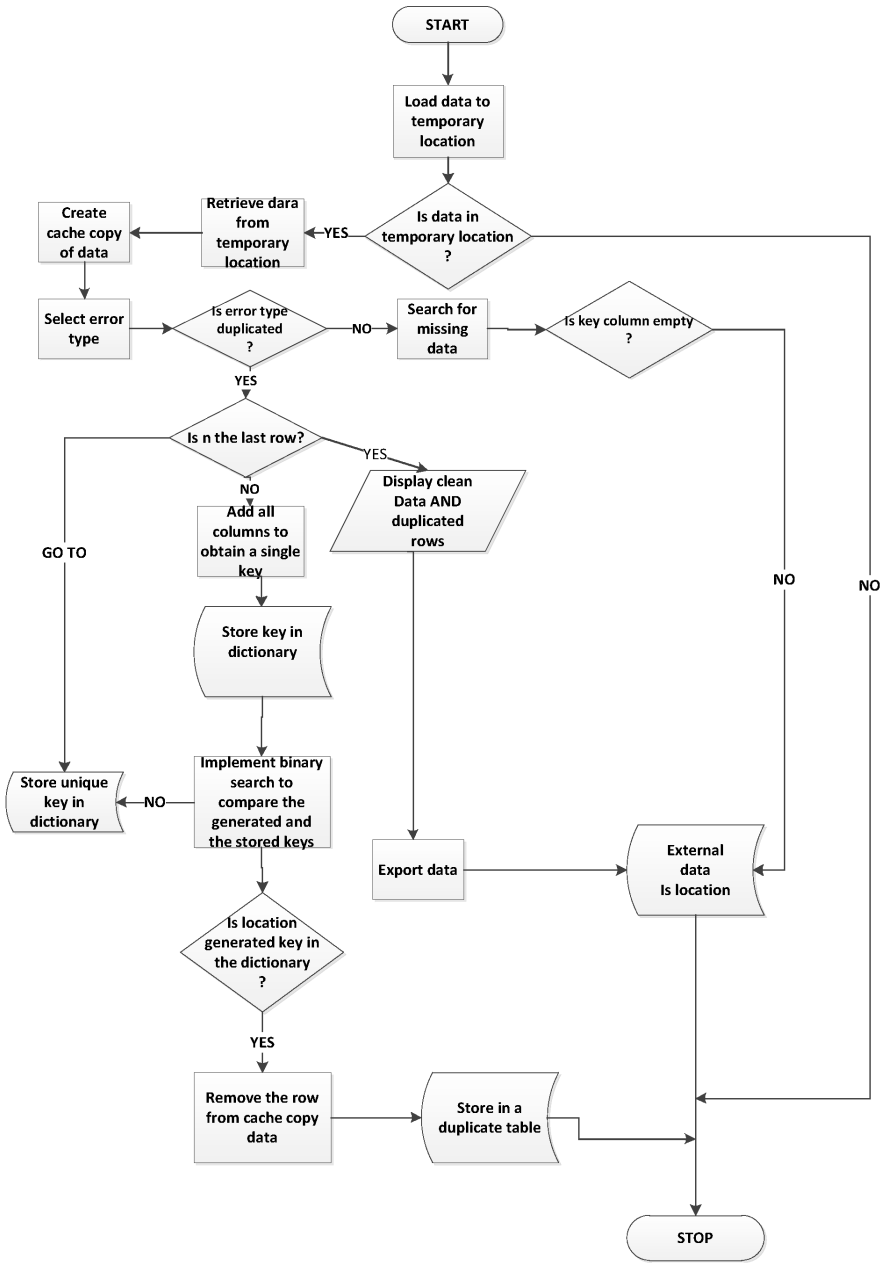

Fig. 5. Flowchart representation

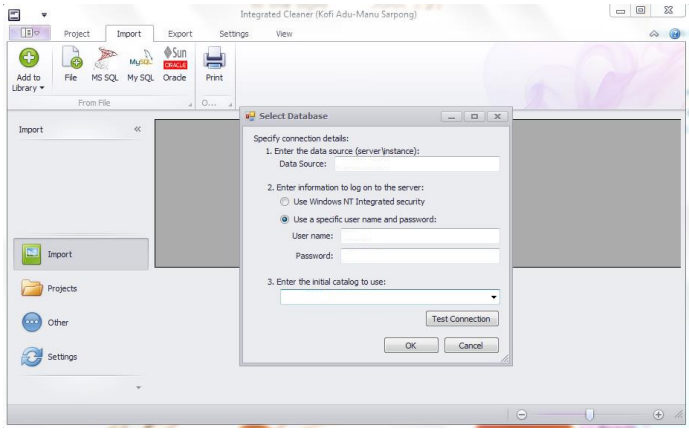

Fig. 6. Interface for Connecting to a Database

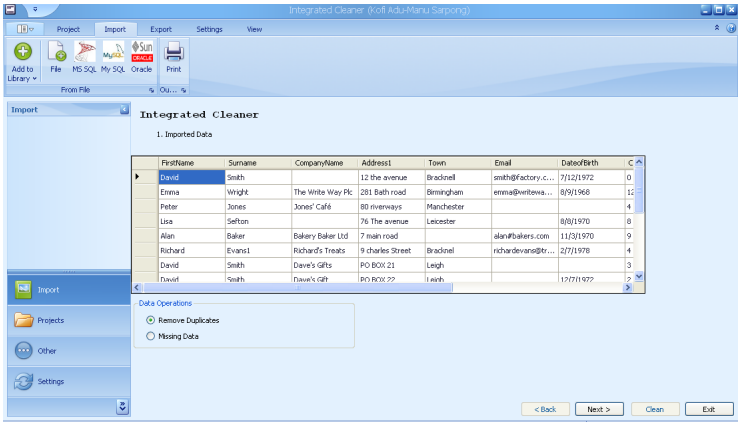

Fig. 7. Interface for Cleansing Error

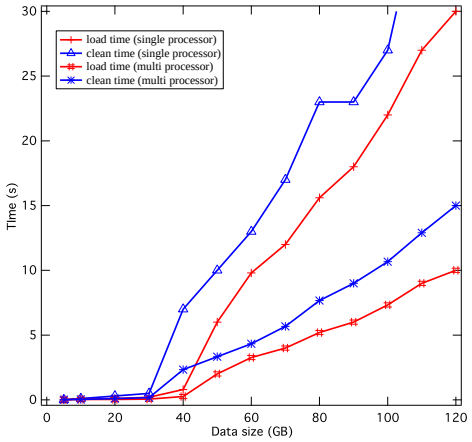

Fig. 8. System testing to remove duplicates

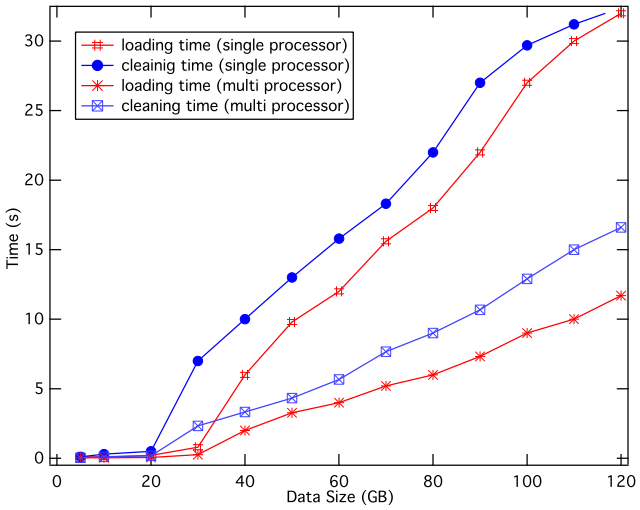

Fig. 9. System Testing to Remove Missing Data

6 Conclusions

In this paper, we propose a data cleansing tool to remove duplicates and missing data fields from databases and text files by implementing a simple but efficient algorithm. We also provide a data cleansing framework, a system model, and a system architecture and describe each stage in the data cleansing process. The system is designed with higher flexibility and can be scaled to add other errors by simply integrating it in the programming logic due to the usage of object-oriented programming tools. The introduction of duplicated records in text files and databases, missing datasets, and other errors have adverse effects on strategic decision making by business owners and the company as a whole. Research has shown that *dirty* data causes a lot of inconsistencies in data reporting and analyzes which causes the organizations to lose goodwill, money, and also breakdown customer relationships and satisfaction. Several research works have been done in the area of data cleansing in the past to overcome the inconsistencies that arise as a result of erroneous data. We have provided a new approach to efficiently perform data cleansing with a simple tool (iCleaner) with enhanced GUI. The algorithms presented and explored in this paper relates to error detection and data cleansing by generating a key and storing it in a dictionary for use during the searching of these records. The designed tool (iCleaner) worked effectively to obtain quality results.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Mller H, Freytag J -C. (). Problems, methods, and challenges in comprehensive data cleansing. Professoren des Inst. Fr Informatik; 2005.

[2] Wayne W-E. Data quality and the bottom line: Achieving business success through a commitment to high quality data. TDWI Report; 2004.

[3] JerryYao D, Sarrab M, Aldabbas H. Three Tier level Data Warehouse Architecture for Ghanaian Petroleum Industry. International Journal of Database Management Systems (IJDMS). 2012;4.

[4] Bradji L, Boufaida M. Open User Involvement in Data Cleaning for Data Warehouse Quality. International Journal of Digital Information and Wireless Communications. 2011;536-544.

[5] Hellerstein J-M. Quantitative data cleaning for large databases. United Nations Economic Commission for Europe (UNECE); 2008.

[6] Karr A-F, Sanil A-P, Banks D-L. Data quality: A statistical perspective. Statistical Methodology. 2006;137-173.

[7] Adu-Manu K-S, Arthur J-K. Analysis of Data Cleansing Approaches regarding Dirty Data-A Comparative Study. International Journal of Computer Applications; 2013.

[8] Krishnan Sanjay, Daniel Haas, Michael J. Franklin, Eugene Wu. "Towards reliable interactive data cleaning: A user survey and recommendations." In HILDA@ SIGMOD. 2016;9.

[9] Hoshino A, Nakayama H, Ito C, Kanno K, Nishimura K. Leveraging the Common Cause of Errors for Constraint-Based Data Cleansing. In Trends and Applications in Knowledge Discovery and Data Mining. Springer International Publishing. 2015;164-176.

[10] Frber C. Data Quality. In Data Quality Management with Semantic Technologies. Springer Fachmedien Wiesbaden. 2016;20-55.

[11] McKelvey N, Curran K, Toland L. The Challenges of Data Cleansing with Data Warehouses. In Effective Big Data Management and Opportunities for Implementation IGI Global. 2016;77-82.

[12] Raman V, Hellerstein J-M. Potter's wheel: An interactive data cleaning system.VLDB. 2001;381-390.

[13] Sarpong KAM, Davis JG, Panford JK. A Conceptual Framework for Data Cleansing-A Novel Approach to Support the Cleansing Process. International Journal of Computer Applications. 2013;77(12).

[14] Cali A, Calvanese D, De Giacomo G, Lenzerini M. Data integration under integrity constraints. In Seminal Contributions to Information Systems Engineering. Springer Berlin Heidelberg. 2013;335-352.

[15] Kopcke H, Rahm E. Frameworks for entity matching: A comparison. Data and Knowledge Engineering. 2010;69(2);197-210.

[16] MLee ML, Ling TW, Low WL. IntelliClean: a knowledge-based intelligent data cleaner. ACM. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. 2000;290-294.

[17] Krishnan S, Wang J, Franklin MJ, Goldberg K, Kraska T, Milo T, Wu E. Sampleclean: Fast and reliable analytics on dirty data. IEEE Data Eng. Bull. 2015;38(3):59-75.

[18] Kimball, R. Dealing with dirty data. DBMS; 1996.

[19] Mohan S, Willshire MJ, Schroeder C. DataBryte: A Proposed Data Warehouse Cleansing Framework. In IQ. 1998;283-291.

[20] Hamad M-M, Jihad A-A. An Enhanced Technique to Clean Data in the Data Warehouse. In Developments in E-systems Engineering.IEEE. 2011;306-311.

[21] Mallach E. Decision support and data warehouse systems. Irwin/McGraw-Hill; 2000.

[22] Nemani R-R, Konda R. A framework for data quality in data warehousing. In International United Information Systems Conference. Springer Berlin Heidelberg. 2009;292-297.

[23] Vassiliadis P, Vagena Z, Skiadopoulos S, Karayannidis N, Sellis T. ARKTOS: towards the modeling, design, control and execution of ETL processes. Information Systems 2001;26(8):537-561.

[24] Kim W, Choi B-J, Hong E-K, Kim S-K, Lee D. A taxonomy of dirty data. Data Mining and Knowledge Discovery. 2003;81-99.

[25] Adu-Manu K-S, Arthur J-K. A Review of Data Cleansing Concepts-Achievable Goals and Limitations.International Journal of Computer Applications; 2013.

[26] Herbert K-G, Wang J-T. Biological data cleaning: a case study. International Journal of Information Quality. 2007;60-82.

[27] Hernndez M-A, Stolfo S-J. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discover. 1998;9-37.

[28] Naumann, F. Quality-driven query answering for integrated information systems. Springer Science and Business Media. 2002;2261.

[29] Galhardas H, Florescu D, Shasha D, Simon E. Ajax: An Extensible Cleaning Tool. In: ACM Sigmod Record. 2000;29:590.

[30] Krishnan S, Wang J, Wu E, Franklin MJ, Goldberg K. Activeclean: Interactive data cleaning while learning convex loss models. arXiv preprint arXiv:1601.03797; 2016

[31] Pulla VSV, Varol C, Al M. (). Open Source Data Quality Tools: Revisited. In Information Technology: New Generations. Springer International Publishing. 2016;893-902.

[32] Khayyat Z, Ilyas IF, Jindal A, Madden S, Ouzzani M, Papotti P, Quian-Ruiz JA, Tang N Yin S. May. Bigdansing: A system for big data cleansing. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM.2015;1215-1230.

[33] Shaik S, Nalamothu NMR. Automatic Data Cleansing of Incorrect Citynames Using LCS Algorithm. International Journal. 2016;4(1).

[34] Wang R-Y, Storey V-C, Firth C-P. A framework for analysis of data quality research. IEEE transactions on knowledge and data engineering. 1995;623-640.

[35] Strong DM, Lee YW, Wang RY. Data quality in context. Communications of the ACM. 1997;40(5);103-110.

[36] Lin J-H, Haug P-J. Exploiting missing clinical data in Bayesian network modeling for predicting medical problems. Journal of Biomedical Informatics. 2008;1-14.

[37] Jonathan I, Marcus M-A. Data Cleaning: Beyond Integrity Analysis. Division of Computer Science; 2000.

[38] Rahm E, Do H-H. Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. 2000;20003-13.

[39] Manning A. Cleansing Your Data. In Databases for Small Business. Apress. 2015;145-165.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/16796