A Classification of Data Quality Assessment and Improvement Methods

Philip Woodall*

Institute for Manufacturing, Department of Engineering, University of Cambridge, 17 Charles Babbage Road, Cambridge, United Kingdom

E-mail: phil.woodall@eng.cam.ac.uk
*Corresponding author

Martin Oberhofer

IBM Germany Research and Development, Schoenaicherstrasse 220, 71032, Boeblingen, Germany

E-mail: martino@de.ibm.com

Alexander Borek

IBM Global Business Services, Hollerithstraße 1, 81829 Munich, Germany

E-mail: alexander.borek@de.ibm.com

Abstract: Data quality (DQ) assessment and improvement in larger information systems would often not be feasible without using suitable "DQ methods", which are algorithms that can be automatically executed by computer systems to detect and/or correct problems in datasets. Currently, these methods are already essential, and they will be of even greater importance as the quantity of data in organisational systems grows. This paper provides a review of existing methods for both DQ assessment and improvement and classifies them according to the DQ problem and problem context. Six gaps have been identified in the classification, where no current DQ methods exist, and these show where new methods are required as a guide for future research and DQ tool development.

Key Words: Data quality, information quality, DQ assessment methods, DQ improvement methods, DQ software tools, DQ assessment, DQ improvement automated DQ tools, automated DQ software.

1 Introduction

Data quality (DQ) assessment and improvement are the two essential stages used to detect and eliminate data problems in organisational information systems (Batini et al., 2009). The act of assessing DQ, to detect what quality problems exist, informs the improvement phase where the relevant data is actually modified to the correct values (Woodall, Borek and Parlikad, 2013). Both DQ assessment and improvement in larger information systems would often not be feasible without using suitable "DQ methods", which we define as: algorithms that can be automatically executed by computer systems to detect and/or correct problems in datasets. Examples include methods such as "column analysis", which can automatically compute statistics such as the number of missing values in a column of data. Many of the current DQ software tools that implement these algorithms can be applied to relational databases that are in use in the majority of organizations today, and automated DQ methods for databases are the focus of this paper.

Performing DQ tasks manually, in many cases, is infeasible (McKenna, 2011), and manual inspection and modification of data is very time-consuming when compared to the speed at which a software program can execute (Maletic and Marcus, 1999). Organisations therefore rely on automated DQ methods, and many vendors, large and small, have been providing these for many years. The current trends indicate that the volume of data is increasing at staggering rates ((Manyika et al., 2011) (LaValle et al., 2011), and see, for example, (Finnegan, 2013)), and therefore we can expect that automated DQ methods will become even more important in the future. Furthermore, as the data grows, not only will DQ automation be a necessity, but also new ways of automating methods for DQ assessment and improvement will be required. This paper therefore aims to identify gaps where no automated DQ methods exist for particular DQ problems, and guide future DQ research and industrial tool development before organisations become overwhelmed with their data. To accomplish this, we provide a review and classification of the different DQ methods currently available for DQ assessment and improvement according to the types of DQ problems they can address. Six gaps have been identified in our classification and these indicate the cases where no current automated DQ methods exist for a particular problem. In developing our classification, we discuss the applicability of each DQ method to each problem by giving an illustrative example. We define and use the more abstract term of "DQ method", as above, rather than referring to software tools because our aim is not to focus on which software tool is best, but rather to identify gaps where new individual algorithms are needed. Moreover, existing DQ software tools often encompass a number of DQ methods, and the focus on DQ methods therefore ensures that one method is compared to one DQ problem in our classification.

The rest of the paper is structured as follows: first, we introduce and describe an existing set of DQ problems that we used to populate our classification followed by a list of DQ methods currently used in different DQ tools. We then review other work on DQ method classifications before presenting how we developed our classification and the classification itself. The paper ends with a discussion of the gaps in the classification and a conclusion with an outlook for future research.

2 Data Quality Problems

There are many types of DQ problems in organizations and several researchers have investigated DQ problems and root causes (Oliveira, Rodrigues and Henriques, 2005; Eppler, 2006; Huang, Lee and Wang, 1999; Pipino, Lee and Wang, 2002).

	Data Perspective	**User Perspective**
Context-independent	Spelling error Missing data Incorrect value Duplicate data Inconsistent data format Syntax violation Violation of integrity constraints Heterogeneity of measurement units Existence of synonyms and homonyms	Information is inaccessible Information is insecure Information is hardly retrievable Information is difficult to aggregate Errors in the information transformation
Context-dependent	Violation of domain constraints Violation of organization's business rules Violation of company and government regulations Violation of constraints provided by the database administrator	The information is not based on fact Information is of doubtful credibility Information presents an impartial view Information is irrelevant to the work Information is incomplete Information is compactly represented Information is hard to manipulate Information is hard to understand Information is outdated

Table 1: A grouping of DQ Problems (extended from (Ge and Helfert, 2007))

DQ problems can be grouped into problems that exist independently of a specific context, e.g. spelling errors and duplicate data, and problems that depend on the context of use, e.g. violation of company and government regulations (Ge and Helfert, 2007). Furthermore, these problems can be seen from a user perspective, which are the problems recognized by an information consumer, or a data perspective, which exist independently of a particular user perspective.

In order to keep the scope of this review as concise as possible, this research focuses on the data perspective problems for the context independent DQ problems identified in (Ge and Helfert, 2007) (see Table 1). These types of DQ problems are the most appropriate for automated tools because they are independent of context, which often requires human intervention to establish the correct course of action. The following paragraphs provide a brief definition for each data perspective and context independent DQ problem in the context of our research.

Spelling errors occur when data values have been misspelt and missing data occurs when the data is physically not present in a field, record, table etc. Incorrect values are those that aim to represent some real-world property, but refer to this property inaccurately; for example, a person's date of birth could be recorded as some other date than their correct date of birth. Duplicate data problems occur when rows are duplicated or when schemas

contain redundancies (that is, specify duplicate attributes). Inconsistent data format problems occur when two or more semantically equivalent data values have different representations. Syntax violation problems occur when a pre-specified format has been assigned to an attribute and a data value for this attribute does not adhere to this format (this includes the incomplete data format and text formatting DQ problems in the original classification in (Ge and Helfert, 2007)). Problems with violations of integrity constraints arise when data values do not adhere to pre-specified database integrity constraints; we also therefore include unique value violations, rather than have these as a separate problem, because unique value violations are one type of database integrity constraint. Note that, despite its position original classification in (Ge and Helfert, 2007), we treat outdated data to be a both a context dependent and user perspective problem because whether data is out of date depends on who it is intended for and the purpose it is used for.

Since using the DQ problems in Table 1, we noticed that two other distinct data-perspective and context independent-data DQ problems, mentioned in the taxonomy in (Oliveira, Rodrigues and Henriques, 2005), also exist, and hence these have been included in our review and have been appended to the bottom of the list in Table 1. Heterogeneity of measurement units refers to the situation where one value could be recorded in millimetres and the same value for a different instance could be recorded in centimetres; both values may be correct, but they are only comparable when the appropriate conversion has been applied. The existence of synonyms and homonyms occurs when synonyms/homonyms are used for values in different records, and this also includes the use of acronyms. For synonyms, two different records could use different synonyms for a country name, for example, “USA” and “United States of America”.

3 DQ Assessment and Improvement Methods

To obtain a list of DQ methods we reviewed the existing software tools for both DQ assessment and improvement and extracted the different methods provided within these tools. The landscape of DQ software tools is regularly reviewed by the information technology research and advisory firm Gartner, and we used their latest review to scope the search for DQ tools from which to extract DQ methods (Friedman, 2012). The list of DQ tools reviewed is as follows:

- SAS dataflux
- Informatica
- Trillium software
- SAP
- IBM
- Pitney Bowes Software
- Oracle
- Datactics
- DataMentors
- RedPoint-DataLever
- Uniserv
- Innovative Systems
- Human Inference
- Talend
- Information Builders/iWay
- Ataccama

To perform the extraction of methods, we reviewed the actual tool (for those that were freely available) and any documentation of the tool including information on the organizations’ websites. We also augmented this review with a general review of DQ literature that describes DQ methods, and have cited the relevant works in our resulting list of DQ methods in the following section. Once we had reviewed each tool and

extracted the DQ methods, the methods were validated (for completeness and validity) by an expert with 10 year's practitioner's experience of current practices in the data quality industry. The resulting methods are described in the following two subsections and have been split according to whether they are for DQ assessment or improvement.

DQ Methods for Assessment

As noted before, the aim of DQ assessment is to inspect data to determine the current level of DQ and the extent of any DQ deficiencies (Woodall, Borek and Parlikad, 2013). The following DQ methods, obtained from the review of the DQ tools above, support this activity and provide an automated means to detect DQ problems.

Column analysis typically computes the following information: number of (unique) values and the number of instances per value as percentage from the total number of instances in that column, number of null values, minimal and maximal value, total and standard deviation of a value for numerical columns, median and average value scores, etc. (Olson, 2003). In addition, column analysis also computes the inferred type information. For example, a column could be declared as a 'string' column in the physical data model, but the values found would lead to the inferred data type 'date'. The frequency distribution of the values in a column is another key metric which can influence the weight factors in some probabilistic matching algorithms. Another metric is format distribution where only 5 digit numeric entries are expected for a column holding German zip codes. Some DQ profiling tools (for example, Talend profiler) differentiate between analyses that are applicable to a single column compared to a "column set". A column set analysis refers to how values from multiple columns can be compared against one another. For this research, we include this functionality within the term "column analysis".

Cross-domain analysis (also known as functional dependency analysis in some tools) can be applied to data integration scenarios with dozens of source systems (Lenz and Shoshani, 1997). It enables the identification of redundant data across tables from different, and in some cases even the same, sources. Cross-domain analysis is done across columns from different tables to identify the percentage of values that are the same and hence indicates whether the columns are redundant.

Data verification algorithms verify if a value or a set of values is found in a reference data set (Maydanchik, 2007); these are sometimes referred to as data validation algorithms in some DQ tools. A typical example for automated data verification is checking whether an address is a real address by using a postal dictionary. It is not possible to check if it is the correct address, but these algorithms verify that the address refers to a real, occupied address. The results depend on high quality input data. For example, verification against the postal dictionary will only produce good results if the address information has been standardized (a method described in the following DQ improvement section).

Domain analysis can be applied to check if a specific data value is within a certain domain of values (Olson, 2003). A domain can be a lookup table containing a series of values, some other pre-defined set of values or range conditions where values have to be within certain boundaries. Domain analysis and data verification are very similar. The

key difference between these is that the domain is just a series of values that are not necessarily the only real possible values. It is possible to do a domain analysis to check if all values for a zip code are between 10000 and 99000, but not all values between these numbers may be valid zip codes in Germany. Hence, data verification is needed to check whether a particular zip code is a real in-use zip code using the official address dictionary in Germany.

Lexical analysis is usually applied to columns containing 'string' values with the intent to discover sub-components of unstructured content. There are basically two major approaches: rule-based and supervised-model based techniques (Borkar, Deshmukh and Sarawagi, 2001; Agichtein and Ganti, 2004). For a field containing "Müller , Christian", lexical analysis would, for example, identify "Christian" as a first name based on match against a dictionary, the gender of the first name to be male and "Müller" to be the last name. Furthermore, phonetic algorithms, such as, NYSIIS or SOUNDEX (Taft, 1970; Patman and Shaefer, 2001), are leveraged by lexical analysis to discover equivalent phonetic representations, such as "Müller" is equivalent to "Mueller". Lexical analysis is a technique often applied to name fields, address fields and text fields for product information.

Matching algorithms (also referred to as record-linkage algorithms or merge-purge algorithms when considered with data consolidation algorithms described in the next section) are used to identify duplicates such as two customer records that refer to the same customer (Gu et al., 2003). These algorithms typically operate on rows of data within a table and are often the fundamental components of master data management (MDM) solutions. These algorithms are linked to the data consolidation algorithms described in the next section. There has been a significant amount of research on matching algorithms, starting with the first formalisation of the method in 1969 (Fellegi and Sunter, 1969). More recent work includes approaches that significantly reduce the runtime of matching, see for example, the "Sorted Neighbourhood Method (SNM)" (Hernández and Stolfo, 1995).

Primary key and foreign key analysis (PK/FK analysis) is applied to columns in two or more tables to detect whether or not the analyzed columns are good candidates for a PK/FK relationship that is not explicitly defined in the data model. For example, a technique for single and multi-column PK/FK discovery can be found in (Li et al., 2006) and an efficient method avoiding limitations by implementing PK/FK discovery with a pure SQL approach can be found in (Bauckmann, Leser and Naumann, 2006).

Schema matching detects when two attributes are semantically equivalent. It offers help to the data modeller using appropriate tools using database schema matching algorithms, which are either schema-based matchers, instance-level matchers or hybrid approaches (Rahm and Bernstein, 2001). However, on large real-world data models with a few thousand or more attributes, current schema matching algorithms are often not very effective, although, additional improvements have been proposed in (Byrne et al., 2009; Peukert, Eberius and Rahm, 2011).

Semantic profiling (also referred to as "business rule analysis" in some DQ profiling tools) is used to express rules on data, and it measures the compliance of the data against these specified rules (Chiang and Miller, 2008). For example, a rule for the columns age

and profession could be: if age < 18 then profession='child', which specifies that the attribute value for profession should be 'child' when the attribute value for age is less than 18. Rules can define any mathematical relationship between numeric data values.

DQ Methods for Improvement

Once DQ assessment methods have detected the current problems, it is the improvement methods that actually correct the data. The following DQ improvement methods can be applied to automatically correct data in different cases.

Data standardization algorithms are used to convert data values into their standardized representation. These algorithms might replace "Aktiengesellschaft" with "AG", "Street" with "St.", etc. Data standardization is often applied in the domain of master data which has high requirements on data quality and common techniques include name standardization, address standardization and product standardization. Tools with the capability of address standardization typically provide country-dependent rule sets and dictionaries because the correct structure of an address differs from country to country. For instance in (Kothari et al., 2010) a new address standardization technique for countries such as India is shown which is able to deal with substantial variants in address structures within a country. Also for addresses, (Guo et al., 2009) proposes latent semantic association instead of the traditional rule and dictionary-based approaches for data standardization. The data standardization step is typically applied before matching and data consolidation algorithms are applied to match records with the intent to find duplicates.

Data enrichment complements the known attributes of a record with additional attributes from trusted internal or external sources. For example, organisations may wish to enrich their customer master data with legal information about enterprises. In the product domain, some companies subscribe to the Global Data Synchronization Network (GDSN) managed by the GS1 non-profit organization. Through the GDSN, companies get access to product classification information, unique product identification numbers (e.g. GPC, GTIN, etc.), etc. which is used to enrich product information.

Data consolidation (also known as data merging or data de-duplication algorithms) is the process of merging two or more duplicate records into one record. This reconciliation of data can be done manually or automatically and it uses the results from the matching algorithms (described in the assessment methods section) that have flagged records that need to be merged. There are two distinct methods available for merging duplicates: record-level survivorship or attribute-level survivorship. The first identifies, from a set of duplicates, one record as survivor and all others are marked as non-survivors and linked to the survivor. If attribute-level survivorship is applied, then for each attribute the decision about which value is used is made for each attribute individually. An excellent survey on this topic can be found in (Bleiholder and Naumann, 2008) and new techniques can be found in (Bleiholder et al., 2010). Again, this technique is quite often applied to master data. Based on business requirements the duplicates are reconciled either with record-level survivorship or attribute-level survivorship. Note that data consolidation can be done fully automatically as well as through Data Stewards in a data stewardship processes.

Data integration (also referred to as Extract-Transform-Load (ETL)) is a broad field of methods which are used to move data from one system to another. The first stage is to extract the data from the source system(s), and then transforms usually need to be applied before loading the data into the target system(s). Within data integration, and especially the transformation stage of ETL, the methods include: transcoding values (for example, 'm' to 'male' and 'f' to 'female'), data type conversion (e.g. 'string' to 'timestamp'), data reformatting (for example, date formats from. yyyy/mm/dd to dd/mm/yyyy and including regular expression (regex) string replacement), unique key generation, and auto completion of originally 'null' values that cannot be null in the target system. The latter is an example of a data rule, which can be developed to add new data or change existing data based on a conditional rule. For example a data rule could be if col1='X' and col2 ='Y' then col3='Z', which conditions the data based on business requirements. Typical use cases for ETL algorithms are feeds for data warehouses or master data management systems.

4 Existing Classifications of Data Quality Methods

The requirement for organizations to assess and improve their DQ, has led to similar research efforts that aim to guide organizations in their selection of DQ tools. For instance, related research specifies useful criteria for the selection of DQ tools based on the functionality of the tools (Goasdoué et al., 2007). However, this work does not indicate the specific DQ problems which are addressed by these tools. Our research differs in this respect, and one of the dimensions of our analysis is the DQ problems and how each DQ method addresses them. In the same subject area, Gartner research takes a different approach and provides a "Magic Quadrant" to guide organizations in their selection of DQ tools. The quadrant indicates which tools fit into the following categories: market leaders, challengers, niche players and visionaries (Friedman, 2012). In a survey of DQ tools, Barateiro and Galhardas divide DQ tools by their general functionality (e.g. debugging capabilities and the ability to extract from different data sources etc.) and also specify whether the tools are capable of finding problems that relate to multiple or single rows (Barateiro and Galhardas, 2005). In this latter sense, the work is based partly on an existing taxonomy of DQ problems (Oliveira, Rodrigues and Henriques, 2005) and our research extends this to specify how specific DQ methods (rather than DQ tools) fit into all the elements of the taxonomy, not just multiple or single rows.

5 Classifying DQ Assessment and Improvement Methods

An existing taxonomy (see (Oliveira, Rodrigues and Henriques, 2005)) was used as part of our classification of DQ methods to problems. This consists of elements at various levels of granularity that relate to the well-known relational database structure which includes: attributes (fields or columns), rows (records or tuples), tables (relations) and the database (multiple tables). Furthermore, the taxonomy also includes a level that relates to multiple databases. The different elements of the taxonomy are shown in Table 2, which includes the taxonomy element (an acronym of the element), the name of the element, and a mapping requirement. The mapping requirement (not described in the original taxonomy in (Oliveira, Rodrigues and Henriques, 2005)) is used for this work to aid the mapping of the DQ methods into an element of our classification for a particular DQ

problem. This mapping requirement specifies what level of data the DQ method needs in order to detect or correct a DQ problem. For example, for the domain analysis method to determine whether there is an incorrect value, it only needs to consider whether the value of one attribute lies in a domain; thus, it can be classified within the Single Attribute Single Tuple (SAST) element for the incorrect value DQ problem.

Taxonomy Item	**Name**	**Mapping Requirement**
SAST	Single Attribute Single Tuple	One attribute to be compared to external information
SAMT	Single Attribute Multiple Tuples	Comparing multiple rows using one attribute
MAST	Multiple Attributes Single Tuple	Comparing multiple attributes in one row
SR	Single Relation	Comparing multiple attributes between multiple rows in a single relation
MR	Multiple Relations	Comparing multiple attributes between multiple rows in multiple relations, for instance, by using the primary/foreign key links between relations.
MDS	Multiple Data Sources	Comparing data from different sources, e.g. multiple data bases, possibly with different data schemas and semantics

Table 2: Classification Mapping Requirements

Note that there is no reason why the domain analysis method could not be applied multiple times (to each value in a row and for different attributes) to detect incorrect values. This is, however, just a repeated application of domain analysis to different values. Other DQ methods can meet the requirements of multiple taxonomy items depending on the DQ problem they address. For example, semantic profiling, which usually involves checking whether multiple values adhere to a particular relationship, can be used on ‘single attribute multiple tuples’, ‘multiple attributes single tuple’, ‘multiple relations’ and ‘multiple data sources’.

The classification developed in this paper therefore includes both the elements of the taxonomy and DQ problems to demonstrate what problem the DQ method addresses and at what taxonomy level the DQ method needs to be implemented (e.g. single attribute single tuple etc.).

6 A Classification of DQ Methods

Table 3 shows the results of the classification with the taxonomy elements across the top and the DQ problems shown vertically. Each DQ method is placed in the relevant cells with a reference number in brackets. The data quality assessment techniques are in normal font and the data improvement techniques are in italic. For some cells the elements of the taxonomy may not be relevant for a given DQ problem, and in these cases, the cell is shaded grey. For example, a spelling error only relates to a single attribute (and hence only relates to SAST and SAMT) and, by definition, duplicates

cannot occur unless there is a second instance which is a duplicate of the first instance, hence this problem does not occur for SAST. If, however, an example DQ method has been found for a particular problem and taxonomy element, then the cell is populated with the relevant method(s). Otherwise, cells are identified as a gap indicating that no DQ methods are currently available to address the particular DQ problem in the given context.

Gaps in the classification

Six gaps were identified from the resulting classification and each gap is described below; gaps 1 and 2 have been described in the opposite order because gap 1 is an extension of gap 2.

Gap 2: The first gap is for detecting missing data in single relations. Instances of this problem occur when, for example, an organization records an item of stock as a row in a table (the full table is the inventory of stock) and items of stock may be physically present in the warehouse but do not appear in the table as a row. This is a gap since there are no methods that can detect whether data records for physically present entities are missing. The improvement methods depend on the assessment methods and so it is not possible to specify these until assessment methods are proposed.

Gap 1: The problem that this gap relates to occurs when a null value can only be detected by observing other rows. For example, a customer record could have a “subsidiary company” field that links to other customers who they own as a subsidiary company. This field should not be null if another row exists with a subsidiary organisation, but it should be null if there are no other organisations that are subsidiaries. The only way to detect whether this null is valid or not is therefore to inspect other rows.

DQ Problems	***Taxonomy element***					
	SAST	***SAMT***	***MAST***	***SR***	***MR***	***MDS***
Spelling error	(1) Data verification, Lexical analysis, *Data standardization, Regex string replacement*	(2) Column analysis, *Data standardization, Regex string replacement*				
Missing data	(3) Domain analysis, *Data enrichment, Data rules*	**Gap 1**	(4) Semantic profiling, *Data rules*	**Gap 2**	(5) Semantic profiling, *Data rule*	(6) Semantic profiling, *Data enrichment, Data rules*
Duplicate data		(7) Column analysis, PK/FK analysis, *Data consolidation, Data rule*	(8) Semantic profiling, Lexical analysis, *Data standardisation*	(9) Matching algorithms, Schema matching, *Data consolidation*	(10) Matching algorithms, Cross-domain analysis Schema matching, *Data consolidation*	(11) Matching algorithms, Cross-domain analysis, Schema matching, *Data consolidation,*
Incorrect value	(12) Lexical analysis, Column analysis, Domain analysis, Data verification, *Data standardization, Data type conversion, Transcoding*	(13) Column analysis, Semantic profiling, *Key generation*	(14) Semantic profiling, Data verification, *Data rules*	(15) Semantic profiling, *Data rules*	(16) Semantic profiling, Column analysis, *Data rules, Data reformatting*	(17) Semantic profiling, Domain analysis, Column analysis, *Data reformatting*
Inconsistent data format		(18) Column analysis , *Data standardization, Regex string replacement, Data reformatting*	(19) Column analysis, Semantic profiling, *Data rules, Regex string replacement, Data reformatting*		(20) Column analysis, Semantic profiling,, *Data type conversion, Data reformatting*	(21) Column analysis, Semantic profiling, , *Data type conversion, Data reformatting*
Syntax violation	(22) Column analysis, Domain analysis, Lexical analysis, *Regex string replacement, Data reformatting*					
Violation of integrity constraints	(23) Domain analysis, *Data rules*	(24) Column analysis, *Key generation*	(25) Semantic profiling, *Data rules*	(26) Column analysis, *Key generation*	(27) PK/FK analysis, *Data rules*	(28) Semantic profiling, Domain analysis, *Transcoding, Data rules*
Existence of Synonyms and Homonyms		**Gap 3**	**Gap 4**		**Gap 5**	**Gap 6**
Heterogeneity of measure units		(29) Column analysis, Domain analysis, Semantic profiling, *Data rules*	(30) Column analysis, Semantic profiling, *Data rules*		(31) Cross-domain analysis, *Data rules*	(32) Cross-domain analysis, *Data rules*

Table 3: Classification of DQ methods to DQ problems

An example is shown in Table 4. Although, this example is based on a non-normalised structure and, hence, is slightly artificial because if it is normalised the problem manifests itself as the problem described in Gap 1. Assuming that Turbo Motorsport Tuning does not own a subsidiary company, then the subsidiary company field correctly contains null. If however, FZR Motor Vehicle Technologies contained a null in the subsidiary company field, then it would be incorrect, because it should be linked to Turbo Motorsport Tuning.

id	Customer name	Subsidiary company
1	FZR Motor Vehicle Technologies	2
2	Turbo Motorsport Tuning	

Table 4: Example where a null can only be detected by observing other records

One way to detect these types of problems is to use data enrichment to describe which companies are subsidiaries and check the subsidiary field for incorrect nulls where a subsidiary relationship exists and is not recorded. However, there is no individual method to address this problem and it is necessary to build custom code to both detect and correct these types of problems. Furthermore, in this scenario a desired course of action could be to modify the data model and hence, the problem would be as described in Gap 1.

Gap 3: A series of gaps are present in the classification for the detection of the existence of synonyms and homonyms, which are not generally possible to detect with algorithms today. These include gap 2, which describes how synonyms occur between different tuples for one attribute (SAMT). An example is the use of different aircraft part numbers for the same physical part. For a part number attribute, a number in one row may be the vendor part number and another row may contain the part number used by the aircraft manufacturer.

Although there are no specific methods to detect this problem, professionals often build custom logic using the data integration capabilities of the commercial tools to search for synonyms and homonyms. With this, some organisations build enterprise data dictionaries using ontologies, which include a list of synonyms and homonyms for business terms.

Gap 4: An example for gap 3 (albeit almost artificial and a consequence of poor data modelling), which relates to multiple attributes in one tuple (MAST), is a table that stores a column “profession” and a column “job category”, with entries such as:

- Professor, Teacher
- Developer, IT
- Architect, IT
- Teacher, Teacher
- Architect, Construction

In this case, the homonym ‘Architect’, which could refer to an IT or construction architect, could only be detected by looking across two columns. Currently, there are no specific methods to automatically detect homonyms in this case.

Gaps 5 and 6: Both gaps 4 and 5 relate to synonyms and homonyms found in multiple tables in a single database and multiple databases, and these are discussed together because there is little difference between the two cases. A common area for synonyms and homonyms to occur is in the domain of reference data, which is typically stored in lookup tables (in either the same database or a different database). For the synonym problem, the same semantic set of country codes in a lookup table in one system might use an integer-based value set for the lookup values whereas another system might use two-letter values. As a result, the country Germany might be represented with the value '62' in one system and 'DE' in another system indicating that these values are actually synonyms. Since in many cases enterprise applications are customizable regarding lookup tables and their values, a company using multiple instances of the same application might suffer substantially by different customizations of lookup tables per deployed application instance. In one organisation the authors observed this problem, and for roughly 200 countries several thousands different lookup values were customized in several instances of the same application. Not surprisingly, in the data warehousing environment where the operational data from all applications came together, revenue reports by country did not show meaningful results. Similarly, for homonyms, assigning the same meaning to different lookup values across systems can also cause substantial problems.

Discussion of the entries in the classification

For each of the DQ methods in Table 3, the reason for classifying these methods into the particular cells is described. The numbers in braces in the table are used to reference the descriptions.

(1) Using data verification, for spelling errors on a single attribute of a tuple, it is possible to check whether a particular word appears in a dictionary and therefore whether it is spelt correctly. For any particular word, the cases are that it is in the dictionary (and therefore spelt correctly), not in the dictionary because the dictionary is not complete, or not in the dictionary because it is spelt incorrectly. The effectiveness of the method therefore relies on the comprehensiveness of the dictionary. The effectiveness also relies on being able to extract particular words from a field. Therefore, methods such as lexical analysis and data standardisation can be used prior to data verification to ensure that individual words are extracted and passed to the verification algorithm. For correcting the data once spelling errors have been detected, it is necessary to find the correct spelling and replace the old value. Within data integration methods, using a find and replace with regular expressions in the search can be used to fix a particular recurring spelling issue. It is also possible to apply string distance measures (often used in data matching algorithms) to find the "closest" word in a dictionary and then replace the value with this word; survivorship rules (from data consolidation algorithms) can be applied to control this value replacement.

(2) A For spelling errors in a single attribute over all rows, column analysis can be used to identify misspellings by examining inconsistencies between rows. A count of the frequency of values can indicate a spelling error that is not easy to detect manually. For example, in a table of product data, the value "fatscreen" may appear only once in the product name column, while the majority of other rows contain the value "flatscreen" for this attribute. Similar to (1), the same correction methods can be applied. Data

standardisation is especially useful in this case to standardize the spellings of names, addresses, etc. between rows to make them consistent.

(3) By specifying that "null" is not a permissible value in the domain of an attribute, domain analysis can be used on a single attribute in a single row to detect missing data. To correct this problem, a business rule approach, within data integration, can be used to map the null value to a permissible domain value. Data enrichment can also be used to complete null values if the enrichment organisation has information that can be appended to an existing record. For example, an organisation may hold information about a customer, but not have a value that indicates the customer's credit score, which could be obtained from a data enrichment organisation.

(4) For null values that depend on other columns, it is possible to define semantic rules (using the semantic profiling method), such as "If (age > 18) then (profession != null)", which can detect these. This indicates that a person of age 18 must have a non-null value in the profession column. To correct this problem, data rules from the data integration method can be applied to insert the correct value, derived from the rule, in place of the null.

(5) For detecting missing data that requires inspection of multiple tables, one example problem occurs with hierarchical customer objects. A typical customer object in an ERP system could consist of multiple tables in a hierarchical table structure. Sometimes, in a business to business scenario, the customer object in the ERP system is customized so that for each enterprise customer there has to be at least one contact person managing the relationship to that customer. The core customer information in the table is representing the root node of the table hierarchy and the contact person would be in a table being a child node of the root node. Using semantic profiling across the multiple relations (the parent and child table in this case), it is possible to check if for each enterprise customer record in the parent, there is at least one entry in the child table storing the contact person information. If no contact person is found, semantic profiling can define a rule that could flag this as a missing record.

For the example given, either a default contact person record is created and using a data rule used to fill in the missing contact person record or the same is done using existing contact persons. This might be filtered using regional information from the customer object so that there is a default (existing) contact person by region for each customer who has not yet a contact person assigned.

(6) For missing data between multiple systems, assume that customer information is stored in one data source (for example, a master data management system (MDM)) and another data source is used for customer orders. A semantic profiling rule could check if for each active customer in the MDM system, there is at least one order by this customer placed in the last 2 years. Otherwise, the customer status may want to be marked as inactive. Updating the customer status based on such semantic rules is an application of a data enrichment method. In more complex examples, it's likely that complex business data rules updating the data will be applied.

(7) To identify duplicates in different rows, column analysis can be used to create a frequency distribution counting if there are two or more occurrences of values to identify

duplicates. The description of the data model for an attribute might indicate a uniqueness constraint (e.g. a unique index or a primary key). Using column analysis, it is possible to detect whether or not the values across all records for this attributes are unique and thus satisfy this requirement. A PK/FK analysis could therefore also be used with column analysis to flag duplicate values in a column. To correct the problem depends on the nature of the duplicate. It may indicate that the entire record is a duplicate, in which case data consolidation algorithms should be used. However, it may be only the value that is incorrect in which case a specific data rule would be needed to correctly update the data.

(8) In semantic profiling expressions like "If a1 = a2 and a2=a3 and … then record exception" (where a1, a2, a3 are attributes) can be specified to discover if multiple columns in the same row have the same value (i.e. to find duplicates between attributes). In some cases, users enter dummy data like "abc" in all mandatory fields to pass validation checks when they don't have the information available, and this technique is useful to detect this problem. Other cases involve users duplicating values in part of a field such as duplicating the state code in a "country" field as well as having this value in a "state" field, see Table 5. This would require that the country field split into individual words so methods such as lexical analysis and data standardisation could be applied to perform this.

id	State	Country
1	CA	USA, CA
2	FL	USA, FL

Table 5: An example of duplicate values in different fields

(9) For master data management projects, de-duplication of employee, supplier, customer, product, account, etc. records in the database tables is a requirement. Probabilistic matching techniques are a commonly used assessment method, which works on multiple attributes and rows to check for matches. . At the schema level, rather than with the individual values, schema matching can be used to detect when attributes in multiple tables are redundant. Data consolidation is then applied to merge the duplicate records into a single survivor.

(10) Similar to (9) master data can also be duplicated because information like customers often exists in multiple relations. Cross-domain analysis is used across columns of different tables to identify the percentage of overlapping data, which might be an indication that two different tables are duplicates. Similar to (9) data consolidation methods are relevant DQ improvement methods to address duplicates in this context.

(11) Duplicates may also occur between tables in many different organisational databases. As well as being duplicates in a single table and multiple tables in a single system, duplicates can also occur in multiple tables in different systems. The methods at this level are therefore the same as in (10).

(12) To detect incorrect values in a single field, lexical analysis can be applied. For example, lexical analysis could be applied to an address field with the value

"Hirschkopfstrasse 13 71149 Bondorf 07457948953" and it would report that "07457948953", as a token, cannot be recognized as an element of an address and is thus an incorrect value. Also, a column analysis could be applied to check for type discrepancies such as the value "abc" in a column, which according to the metadata should be of data type INT, and, thus, is obviously a wrong value in that column. When a particular domain of values is available for validation of a field, domain analysis could operate on a column containing titles (for example, "Mr.", "Mrs.", "Dr." and "Prof.") and if it finds a value not in the domain, such as "Herr", it would report this value to be incorrect. Another example of domain analysis could be a data integrity constraint on a column storing AGE information that the permissible range of values is between 0 and 140, so that any value outside this range is flagged as incorrect. Data verification could also be applied to determine that a syntactically correct value is not correct if it does not exist in an external data dictionary. In order to correct the data, the data standardisation method can be used to solve the problem with addresses and any case where an incorrect value needs to be extracted from a field containing free text. The data integration method of data type conversion (e.g. STRING to TIMESTAMP), can be used to correct data that is of the wrong type. Also within data integration, the transcoding values method can be used to map any incorrect entries detected by domain analysis to the correct value; for example, the title "Herr" could be mapped to "Mr".

(13) For incorrect values that are detectable by observing the value in a field in multiple rows, one example, is a business decision that requires products should be numbered sequentially by a product number increased by a fixed value or fixed pattern (e.g. 1, 2, 3, 4, 5, … or 000010, 000020, 000030, 000040, …). Using column analysis or semantic profiling can help to discover if data complies to such a number scheme. To correct this problem the key generation method from data integration is applicable to fix this problem.

(14) An example incorrect value detectable from multiple attributes could be a rule such as: if (gender = male AND religion = Roman-Catholic AND profession = Priest) then marital-status =single. Semantic profiling could be used to represent this rule and check whether the data adheres to it. Data verification can also be used to detect whether or not syntactically correct data is valid. For example, if: street = Oxford Street, House Number = 185, Zip code = W1D 3DG, City = London and Country = UK, then this would be a syntactically correct address. Performing data verification against the UK postal address dictionary would flag that all values are correct except for the HOUSE NUMBER for which number 185 does not exist because the largest number in Oxford Street is 157. The data rules method in data integration is used to solve these types of problems.

(15) An example of an incorrect value that is detectable by observing multiple rows and multiple columns is shown in Table 6. In this case the time periods should not be overlapping and, hence, the beginDate value in the second row should be 2/1/2002.

BeginDate	**EndDate**	**ProductsSold**
2/1/2001	1/1/2002	100
2/7/2001	1/1/2003	150

Table 6: An example incorrect value that is detectable by observing multiple row and columns

Semantic profiling can be used to detect this problem by defining a rule that checks each date to see if it is between any other date range specified by the begin and end dates in other rows. To correct the problem, data rules in data integration can be defined to adjust the date so that it follows from the known latest date.

(16) It is possible to check for incorrect values across tables using semantic profiling. For example, a semantic rule could be used to ensure that each sales employee is associated with at least one customer. Also, performing a format analysis as part of column analysis might expose that the syntax of data fields is not consistent across tables (e.g. 12/31/2010 in one table and 31/12/201 in another table). Data rules from data integration techniques can be used to replace the violating values to solve the first problem and format conversions from the data integration technique set fix the problem of the second example.

(17) Some of the examples for (15) like semantic rules across multiple tables can also be used in a scenario where the tables come from multiple sources. Additionally, semantic profiling might be used if complementary data sources are involved to check the correctness of data. Domain analysis against the domain value set of the target system for a field might expose that none of the source systems is using the same domain values for this field. For instance, without transcoding (for example, converting a country code “DE” to “62”) while moving a record from a source system to the target system, the target system would be loaded with a data quality defect, since it is not understanding the value “DE”, being not part of its domain value set for that field. Similarly if another source system uses “GER” for Germany, a different transcoding rule needs to be implemented to map “GER” to “DE” for the records coming from this source system. Column analysis might also be used to detect different formats across multiple sources for dates, etc. Data reformatting conversions from the data integration technique set are usually applied to harmonize formats across records coming from different sources.

(18): Inconsistent formats, for example,German phone numbers, in one column should all be compliant with the format of 0049+area code+phone number. Inconsistencies between the formats in different rows for this column can be detected using column analysis, similar to the previous examples. Either data standardization or data integration techniques such as regular expressions or format conversions can be used to correct inconsistent formats.

(19): A table might contain contact information with multiple fields for phone numbers such as a field for office phone, mobile phone and home phone. If someone uses just one mobile phone and wants to be contacted only with the mobile phone number, then the values are semantically equivalent and should appear in the same format. Column analysis or semantic profiling using for example regular expressions can be used to detect whether or not the format is applied consistently across the columns. Either data standardization or data integration techniques such as regular expressions or format conversions can be used as part of a data rule enforcing the same phone number across office, mobile and home phone to correct inconsistent formats if that’s an option available to a customer.

(20) and (21): In both cases, it could be that tables in the same or multiple data sources store similar information like price information, dates, etc. For example, there might be a

table (in the product development system) with product prices with a field for price which is a decimal type. In an order item table (such as an e-Commerce system) containing the individual items of an order, the price field could be a fixed-length STRING of length 32 with leading, padding whitespaces if the price does not have 31 digits. This and similar issues could be detected with column analysis and/or semantic profiling techniques. If data coming from the development and e-commerce system of the example mentioned requires harmonization before loading into a common target, then reformatting string operations, such as trimming in conjunction with data type conversions, can be used.

(22) Syntax violations can be detected using a number of different methods. For example, dates are often required to adhere to a specific syntax (e.g. dd/mm/yyyy) and violations can be detected using column analysis. Domain analysis can be used to identify erroneous values that do not have the right syntax by flagging any value that is not within the domain (provided that the domain contains no syntax violations). Lexical analysis is able to discover if all tokens in a text are recognized as patterns in a specific format of a domain like addresses, products, etc. Assume an attribute “street” with the following value “274 St. John St.”. Lexical analysis in the parsing step would identify 4 tokens: ‘274’, ‘St.’, ‘John’ and ‘St.’. In the lexical investigation, it classifies the tokens as follows: ‘274’ is identified to be a number, ‘St.’ is identified as a possible street type, ‘John’ is identified as an alphanumeric string possible being a unique name and ‘St.’ is again identified as a possible street type. In the final step applying context sensitive interpretation of the lexical investigation, the conclusion would be that ‘274’ is the house number, ‘St. John’ would be the street name and the last ‘St.’ must be the street type indicator. The output of the lexical analysis would be, in addition to the input column “street”, three columns indicating house number, street name and street type and the string would have not produced any unhandled pattern (a token which the lexical analysis would flag as not understood based on available dictionaries and rules). With the additional three columns applying address standardization is now able to produce much better results. Regular expressions within the integration technique can be used to correct syntax errors by using them with search and replace functions. Reformatting methods can also be used where the correct syntax format is known.

(23) One integrity constraint on a single value is that the primary key cannot be null. This is usually enforced by the database and does not need manifest as a problem; however, the detection of null values was discussed as a separate problem previously. Another relevant integrity constraint is a user defined integrity constraint on a single value. For example an age field may need to be between 16 and 40. Domain analysis can be used to determine if all values for an attribute are within an allowed range. Similar, using domain analysis, it is possible to check if a value in a field is in a given value set of the lookup table supporting this field. Data rules from the data integration technique set can be used to deal with violating values in some instances.

(24) Column analysis can be used to detect if all the values in a column are unique. If the column supposedly only contains unique values, then the number of violations of the uniqueness constraint determines if it is feasible to use key generation from the data integration technique is a viable improvement option.

(25) With semantic profiling, business constraints across multiple attributes can be detected. An example could be selling price = base price * sales tax. Data rules can be

used to fix violations. For example, if there is an erroneous entry of ‘abc’ in the selling price attribute, then a business rule computing base price * sales tax could be used to fix this error (and all other incorrect sales prices).

(26) Column analysis can be applied to a set of fields to detect whether or not they violate a composite primary key constraint which requires unique value combinations across all records in the fields of the composite key. The number of violations of a composite primary key constraint determines if it is more feasible to fix with a key generation approach from the data integration technique set.

(27) Primary/Foreign key analysis can be used to detect if two tables in a parent-child relationship comply with the Primary/Foreign Key constraint for the key columns. Data rules could be used to fix this type of problem to update the values in the relationship between the tables.

(28) Semantic profiling with rules can be used to detect if integrity constraints are complied with for data across sources. For example, a rule can check between data tables, from a sales territory management system and tables from the human resources system, whether only seasoned sales employees, which work at least for 3 years in sales, should have been assigned to customer accounts in the platinum customer segment. Data rules can be applied to fix violations of integrity constraints. For the example given, for any violation found there could be a default sales employee per sales territory assigned to a platinum customer with the necessary experience. During data migration from a source system to a target system, a domain analysis for source records after transcoding can discover if the transcoding has been configured correctly so that the transcoded values indeed are all known by the target system.

(29) The values in a single column could be based on different measurement units and, hence, any pair of values from different records could not be meaningfully compared without a conversion factor. This type of problem can be detected with a column analysis by checking the distribution of values and looking for outliers. Domain analysis could also be used to check that values are in a particular range, however, this method only works when the ranges for likely different measurements do not overlap. Semantic profiling could also be used where common conversion factors (e.g. 1 inch = 25.4mm) are included in the rule to highlight groups of values that differ by this factor. To correct the data, data rules could be defined to find and replace the outliers.

(30) Depending on the data model, a series of measurement values, taken at different places for the same object, may be represented in multiple attributes. Measurement discrepancies in this scenario could be detected with semantic profiling methods and column analysis methods that operate across different attributes (e.g. column set analysis). Similar methods to 29 can be used to correct the problems.

(31 and 32) A problem of different measurement units could exist between tables where in one table millimetres is used and in another, inches is used. Any algorithms that intend to compare these values could not do so meaningfully unless the conversion is known. This problem also extends to the case where the tables are in different systems. In order to detect this problem cross-domain analysis can be used to identify if the values are within the same ranges but have been scaled by a factor (the measurement factor). Data

integration methods, such as data rules, can be used to address the problem; although, if one system needs to change an entire column to a different measurement unit, this may require significant changes to the system. It is also partly a data governance issue because the decision about which system to change and what will be the 'standard' measure that the company uses, needs to be agreed.

7 Discussion and Conclusion

The aim of this research was to provide a review of methods for DQ assessment and improvement and identify gaps where there are no existing methods to address particular DQ problems. Six gaps were identified overall for two DQ problems: missing data and existence of synonyms and homonyms. The first gap relates to the missing data DQ problem, where for example a table is missing a row. In this case the whole relation is needed to determine if any rows are missing (all rows need to be checked) as well as external knowledge of the entity that the missing row(s) should represent. The second gap is closely related to the first and requires an understanding of the relationships between rows in order to detect problems. The remaining gaps relate to the detection of synonyms and homonyms throughout multiple rows, multiple attributes, multiple relations and multiple data sources. For homonyms, this problem often occurs when inconsistent coding schemes are used for domain values in different databases, such as assigning the code '2' and '4' to 'married' and 'single' attribute value options in one database and a different coding scheme in other databases. Clearly, any data that is moved between these databases changes its value inadvertently. New solutions are required to tackle this interesting, but also difficult, problem.

For many of the DQ problems, despite the fact that some problems can be automatically detected and that the correction methods can also be automated, the whole process cannot be carried out automatically without human intervention in most cases. For example, in finding a common spelling error in many different instances of a word, a human is often used to develop the correct regular expression to automatically find and replace all the incorrect instances. So between the application of the automated assessment and improvement methods, there often exists a manual analysis and configuration step.

One limitation of the classification is that it only considers whether a DQ method addresses a DQ problem (for each taxonomy element), and this may not always be absolute in the sense that some DQ methods may be more comprehensive than others—certain DQ methods will have limitations in their use such as execution performance and these have not been captured in the final classification. Future work could therefore address this issue. Future work could also extend the classification by focussing on semi-structured data like XML files and unstructured data. Electronic slide decks (from presentation software) are becoming increasingly important and an analogous classification for these data types is also needed. Lastly, this paper focussed on DQ methods for standard relational databases, and a similar classification for the latest systems, such as "NoSQL" type databases, will be essential as these become more prevalent in organisations of the future.

References

Agichtein, E. and Ganti, V., (2004). Mining Reference Tables for Automatic Text Segmentation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 20–29.

Barateiro, J. and Galhardas, H., (2005). A Survey of Data Quality Tools. *Datenbank Spectrum*, **14**, pp.15–21.

Batini, C. et al., (2009). Methodologies for Data Quality Assessment and Improvement. *ACM Computing Surveys*, **41** (3), p.pp.1–52.

Bauckmann, J., Leser, U. and Naumann, F., (2006). Efficiently Computing Inclusion Dependencies for Schema Discovery. *In Second International Workshop on Database Interoperability, ICDE 06*.

Bleiholder, J. et al., (2010). Subsumption and Complementation as Data Fusion Operators. In *Proceedings of the 13th International Conference on Extending Database Technology*. pp. 513–524.

Bleiholder, J. and Naumann, F., (2008). Data Fusion. *ACM Computing Surveys (CSUR)*, **41** (1), article 1.

Borkar, V., Deshmukh, K. and Sarawagi, S., (2001). Automatic Segmentation of Text into Structured Records. In *ACM SIGMOD Record*. pp. 175–186.

Byrne, B. et al., (2009). Scalable Matching of Industry Models – A Case Study. In *Proceedings of the International Workshop on Ontology Matching, OM*.

Chiang, F. and Miller, R.J., (2008). Discovering Data Quality Rules. *Proceedings of the VLDB Endowment*, **1** (1), pp.1166–1177.

Eppler, M., (2006). *Managing Information Quality: Increasing the Value of Information in Knowledge-Intensive Products and Processes*, Springer-Verlag New York Inc.

Fellegi, I.P. and Sunter, A.B., (1969). A Theory for Record Linkage. *Journal of the American Statistical Association*, **64** (328), p.pp.1183.

Finnegan, M., (2013). Boeing 787s to Create Half a Terabyte of Data Per Flight, Says Virgin Atlantic. *Computerworld UK*.

Friedman, T., (2012). *Magic Quadrant for Data Quality Tools*,

Ge, M. and Helfert, M., (2007). A Review of Information Quality Research. In *Proceedings of the 12th International Conference on Information Quality*.

Goasdoué, V. et al., (2007). An Evaluation Framework for Data Quality Tools. In *Proceedings of the International Conference on Information Quality*.

Gu, L. et al., (2003). *Record linkage: Current practice and future directions*, CSIRO Mathematical and Information Sciences Technical Report.

Guo, H. et al., (2009). Address Standardization with Latent Semantic Association. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. New York, NY, USA: ACM, pp. 1155–1164.

Hernández, M.A. and Stolfo, S.J., (1995). The Merge/Purge Problem for Large Databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*. San Jose, California, United States, pp. 127–138.

Huang, K.T., Lee, Y.W. and Wang, R.Y., (1999). *Quality Information and Knowledge Management* 1st ed., Prentice Hall.

Kothari, G. et al., (2010). Transfer of Supervision for Improved Address Standardization. In *2010 20th International Conference on Pattern Recognition (ICPR)*. pp. 2178–2181.

LaValle, S. et al., (2011). Big Data, Analytics and the Path from Insights to Value. *MIT Sloan Management Review*, **52** (2), pp.21–32.

Lenz, H.J. and Shoshani, A., (1997). Summarizability in OLAP and Statistical Data Bases. In *Proceedings of the Ninth International Conference on Scientific and Statistical Database Management*. Olympia, Washington, pp. 132–143.

Li, F. et al., (2006). *Authenticated Index Structures for Aggregation Queries in Outsourced Databases*, Boston University Computer Science Department.

Maletic, J.I. and Marcus, A., (1999). Progress Report on Automated Data Cleansing. *The Department of Mathematical Sciences Division of Computer Science, The University of Memphis, Memphis, TN*, pp.13.

Manyika, J. et al., (2011). Big Data: The Next Frontier for Innovation, Competition, and Productivity. *McKinsey Global Institute*, p.pp.1–137.

Maydanchik, A., (2007). *Data Quality Assessment*, Technics Publications LLC.

McKenna, B., (2011). Data Quality Improvement Impeded by Lack of Automation. *Computer Weekly*.

Oliveira, P., Rodrigues, F. and Henriques, P., (2005). A Formal Definition of Data Quality Problems. In *Proceedings of the 10th International Conference on Information Quality*. pp. 13–26.

Olson, J.E., (2003). *Data Quality: The Accuracy Dimension*, Morgan Kaufmann.

Patman, F. and Shaefer, L., (2001). Is Soundex Good Enough for You? On the Hidden Risks of Soundex-Based Name Searching.

Peukert, E., Eberius, J. and Rahm, E., (2011). AMC - a Framework for Modelling and Comparing Matching Systems as Matching Processes. In *2011 IEEE 27th International Conference on Data Engineering*. Hannover, Germany, pp. 1304–1307.

Pipino, L.L., Lee, Y.W. and Wang, R.Y., (2002). Data Quality Assessment. *Communications of the ACM*, **45** (4), pp.211–218.

Rahm, E. and Bernstein, P.A., (2001). A Survey of Approaches to Automatic Schema Matching. *VLDB JOURNAL*, **10**, pp.2001.

Taft, R.L., (1970). Name Search Techniques. In *Bureau of Systems Development, New York State Identification and Intelligence System*.

Woodall, P., Borek, A., and Parlikad, A., (2013). Data Quality Assessment: The Hybrid Approach. *Information & Management*, **50** (7), pp.369–382.