AUTOMATING OBJECTIVE DATA QUALITY ASSESSMENT (EXPERIENCES IN SOFTWARE TOOL DESIGN)

(Practice-Oriented Paper)

Sergei Savchenko
Skwork Studio, Canada
sergei@skworkstudio.com

Abstract: The paper discusses the design goals, current architecture and overall experiences gained in the process of building a software tool aiding human analysts in estimating approximate information quality (information accuracy) of an unknown relational data source. We discuss the algorithms and techniques that we found effective. In particular, we discuss the automated reasoning techniques used to identify common data types for which known data quality rules exist and data mining algorithms discovering unknown probable data quality rules. Tool's human interface issues are briefly overviewed. In this regard we discuss methods used for information quality rule scripting and graphical result presentation.

Key Words: Data Quality, Information Quality, Objective Data Quality Assessment, TDQM, Data Type Recognition, Data Mining.

INTRODUCTION

Organizations undertaking the task of cleansing their databases are first faced with the need to reliably estimate the information quality of their existing data. Often this proves to be a very time consuming and costly task due to:

- Size and complexity of an average industrial database
- Large number of people involved in the development, maintenance and use of the database
- Loss or evolution of elements of database's original design over time
- The nature of some data elements such that no strong data quality rules are possible

Several assessment methodologies exist. The subjective data quality assessment method is based on grading various experiences of system's users and deducing measures based on the statistical analysis of responses. Subjective analysis is important in the very beginning and the very end of the process. It is the users who first experience and signal data quality problems and it is the increase in users' satisfaction at the very end of the data improvement process that is the ultimate measure of project's success [9]. Being by definition subjective and hence approximate and imprecise, this method of data quality assessment is not always a satisfactory guide during intermediate technical steps of the process. During these steps, individual root causes, typical errors, data quality rules and cleansing algorithms must be identified and developed.

Objective data quality assessment techniques measure the degree of adherence of the data to particular information quality rules. Among many dimensions of information quality the one easiest to assess using objective techniques is that of information's accuracy [10]. Although the notion of information quality is

much wider than that of information accuracy, we will use those terms interchangeably. Even in the more narrow case of information accuracy, the objective assessment is often limited by multiple practicality factors. In many situations it is much easier to verify correctness of data's syntactic properties rather than its semantic meaning. It is because some semantic information quality rules may not be known at all whereas others may be of the size larger than the database itself. For instance, whereas it could be feasible to verify whether the syntax of customer's address is correct or not, verifying its semantic meaning (i.e.: if this is the address that exists in reality), may be much harder. When it is possible, the process usually involves a lookup against an authoritative address database whose size may be much larger than the size of the database being verified. Even an authoritative address database may not be able to help in the situation where a wrong (perhaps outdated) existing address is attributed to a customer. This last example clearly demonstrates the limits of objective data quality assessment.

All practical syntactic and semantic information quality rules should be a part of the database's original design and subsequent redesigns. On practice, however, documented rules are often of inferior quality (if at all existing), especially in the legacy database situations. A relatively costly effort by analysts to identify the information quality rules may be necessary.

Supplying the analysts with a software tool aiding in information quality rule identification and discovery as well as consecutive objective assessment may simplify and speed-up the process to a considerable extend. Such a software tool may improve identification of the scope and duration of the data improvement project, help to locate which areas of the database require larger share of efforts and also help to document the findings.

In this paper we discuss our experiences in designing and building a prototype for such a software tool. In particular we discuss the question of designing a suitable human user interface with consistent information quality rule scripting facility, visualizing results of information quality assessment in a meaningful way, identifying known and discovering unknown information quality rules. We overview the deductive reasoning framework used for rule identification and inductive data mining framework used to discover unknown frequency rules, associative rules, arithmetic equation rules, regular expressions and context free grammars.

Since this represents an early report of a work in progress, we primarily discuss the design of the developed framework rather than the findings regarding its real-life performance.

DESIGN GOALS

In our effort to develop data quality assessment software tool we start from the following set of goals:

- The tool should identify known data quality rules in a single item, column, row and table contexts
- The tool should attempt automated discovery of unknown, probable data quality rules for the above-mentioned set of contexts
- The user must be able to validate the identified and discovered rules, make changes or introduce new rules
- Data quality estimate should be performed based on the identified, discovered or manually entered rules
- Where possible, the results of the data quality estimate should be graphically visualized
- Multiple iterations of rule editing followed by information quality assessment must be allowed

Data quality rules operating in the context of a single item are such where a particular logical expression or grammar can be stated for the item independently of other items in the row or the entire table. For instance, we normally consider a number between *0* and *100* as valid for representing customer's age. If

this is expressed as a data quality rule, it will only affect the item under consideration.

Data quality rules operating in the row context exist in situations where there is a dependency between several row items. For example, customer's city and customer's telephone area code have an obvious dependency where every city has a limited set of allowed area codes. This constraint can be expressed as a row context data quality rule.

We only consider uniqueness data quality rules for the column and table contexts at this stage. These are the rules allowing us to estimate numbers of probable row duplicates such as in a simple situation where table's primary key is not unique (the column context) or in a more complex situation where the same multi-field address is entered multiple times with slight differences (the table context).

We can also differentiate data quality rules acting in the context of the entire relational database. A referential integrity constraint where a foreign key must relate to an existing primary key is an example of such a data quality rule. We are not considering any database context rules at this stage.

OVERALL TOOL ARCHITECTURE

In the design of a software tool for automated objective data quality assessment we are primarily targeting relational data sources. Since, at this stage, we are not handling rules of the database context, the analysis can be done on one relation at a time. **Figure 1** presents a diagram of the information flow through the application. As the figure demonstrates, our design is modular with the logic representing identified and discovered data quality rules stored in a textual script file. The scripting approach allows the user to quickly modify the rules while, at the same time, documenting the process.

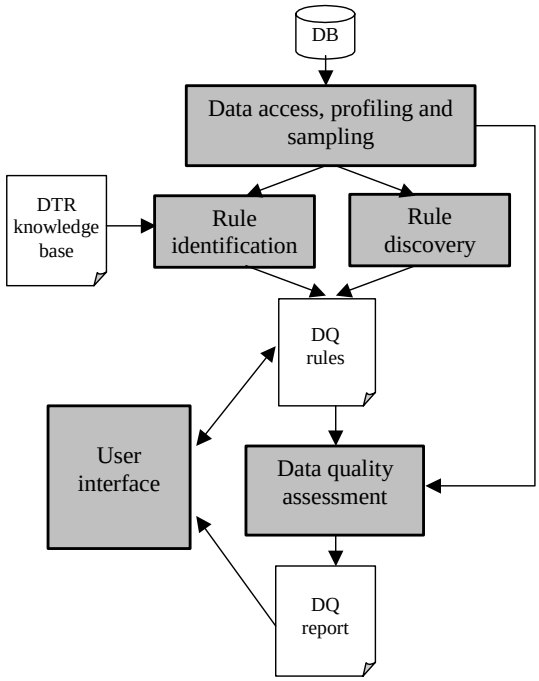

Figure 1: *Information Flow Architecture*

Many industrial databases share significant similarities. For instance, customer information is crucial for

most business operations and hence business databases. Information accuracy constraints are thus well known and understood for many common data types such as name, address, telephone, email fields, etc.

By performing data profiling and deducing semantic type of a data element, we can identify known data quality rules that should apply. For instance by recognizing a certain textual field as the one storing data of a semantic type "U.S. State" we can identify the data quality rule that demands the state to be one of 50 available values.

The rule identification module in our design is responsible for deductive reasoning based on the data type recognition knowledge base and the results of data profiling performed on the data source. The rules are recorded in a textual script file and are read into their memory representation for the use by the deductive reasoning engine. This engine relies on a combination of forward and backward chaining to arrive to conclusions in the form of applicable information quality rules.

The identified rules must be validated by the user. To facilitate this process, the rules are recorded as statements of the internal scripting language whose syntax is similar to that of SQL expressions, or in that matter, expressions of any high level programming language. Note that we use shared syntax for both data type recognition rules and information quality rules. The user can also modify the discovered rules or introduce new rules. Although we are striving to automate the process to the largest possible extend, intimate involvement of the user still appears beneficial since the goal of information quality assessment is not only to obtain a quality measure but also to identify specific kinds of problems.

In many situations, identification of known data quality rules will not succeed. We can nevertheless exploit the regularity properties of databases to attempt deducing probable data quality rules. In the situations where the data is largely correct, we can try to discover clusters of items sharing common syntactic or semantic properties and assume that the outliers (which don't belong to any cluster) likely represent data errors. For instance, by recognizing that a certain column has *n* different values and of those *m* values are very infrequent we may assume that the latter items are erroneous.

It should be mentioned that in many real-life situations it is not just infrequent values that represent potential data quality errors. In fact the most frequent items could be the ones that are problematic. Such a situation may often occur if many customers chose a "lazy" data entry option [7], for instance selecting the first available country of residence from the list or typing "11/11/11" as a date. We consider both outliers and overly dense data clusters as potential data quality issues and allow the user to validate the obtained rules specifying these problems.

Various data mining algorithms in our design are responsible for inductions discovering potential data quality rules. We attempt to discover the following rules:

- Item frequency rules
- Associative rules
- Arithmetic equation rules
- Regular expression patterns
- Context free grammar patterns

Depending on the data profiling results we select those data mining algorithms that have a better chance to succeed. For instance, we will apply regular expression discovery algorithm in the case of a textual field that contains on average one word. Context free grammar discovery will be attempted in the case of a multi-word text field.

Some of the rules that could be discovered clearly belong to the syntactic category. Indeed, both regular expressions and context free grammars serve the purpose of expressing syntax. Discovered arithmetic equation rules can often express some meaning of the data, in particular, some underlining business rule.

Item frequency and associative rules can be viewed as belonging to both syntactic and semantic categories. In some instances an association between two columns of the relation can have syntactic necessity (for instance, an abbreviation of a state that is entered alongside unformatted address field where state information also appears). In other cases, an association can represent some aspect of the meaning of the data, at times even emergent or previously hidden meaning. Thus, although the identification process, as implemented by the tool, primarily finds known syntactic rules (with the exception of uniqueness rules), the discovery procedure can identify certain rules from both categories.

Some of the discovery algorithms can operate on the entire data set. Others (due to their computational complexity) can feasibly operate only on a data sample. As a result, our data access module implements techniques for random sampling.

Once a set of data quality rules is prepared and validated by the user, the tool can compute a data quality estimate by verifying how often the rules were violated in the data. This process involves traversing the entire data set and performing deductive reasoning steps for every data element in order to verify if all data quality rules are upheld.

The evaluation of uniqueness data quality rules that are used to assess percentage of item or record duplicates requires the presence of large hash tables in memory or on secondary storage. Our current implementation maintains the hash tables of limited size in memory and, as a result, we can only estimate approximations for percentages of duplicates.

The computed data quality estimate is presented to the user. Visual presentation of information can often dramatically simplify its analysis. We try, where possible, to supplement textual results with their graphical presentation. We present distributions and histograms for frequency rules; function plots for equation rules; cluster depictions for associative rules. In the cases of regular expressions and context free grammars we may be able to discover multiple syntactic categories. If that has happen, we will visualize histograms presenting value breakdowns by a syntactic category.

By analyzing results, it is entirely possible that a user will decide to modify the set of data quality rules to test new hypothesis or to correct discovered mistakes and inconsistencies.

The iterative process thus continues until the user reaches suitable understanding of data quality issues in the database. The report obtained during the final iteration can thus serve to document the entire activity.

IDENTIFYING KNOWN DATA QUALITY RULES

The problem of identifying applicability of existing data quality rules is tightly related to the problem of identifying the data type of a data entity. Indeed, if the data type is known we can use the data quality rules that were defined for that type.

The data type can often be deduced from the results of data profiling with the help of data type recognition knowledge base. We store the rules of that knowledge base in a textual script file to facilitate possible changes and additions. **Figure 2** illustrates the information flow for the data quality rule identification process.

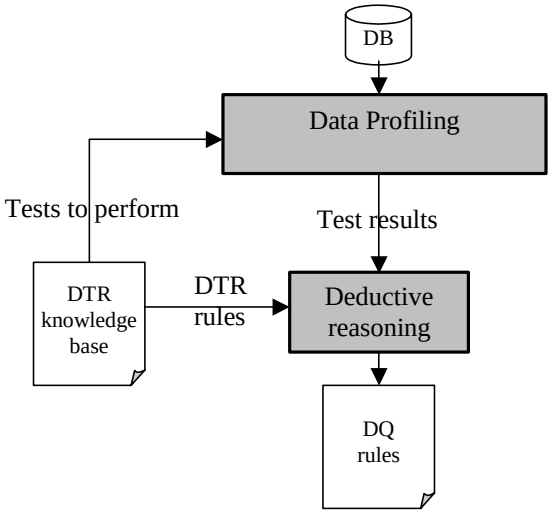

Figure 2: *Identifying known data quality rules*

The basic construct of the scripting language that we use is a rule statement. The rule statement has a form of $E \Rightarrow A$ where E is an expression and A is an assignment statement. The meaning of the rule statement signifies that if the expression is true the assignment will proceed. In the syntax of expressions, we allow a usual assortment of logical, relational and arithmetic operations similar to that of most high level programming languages.

A special provision is made for a set of column matching functions through which extended data profiling can be done. Among many, there are the following functions:

- **range** Computes the percentage of the values that were in a specified range
- **lookup** Computes the percentage of the values that were in a specified lookup table
- **substring** Computes the percentage of the values that contained specified sub-strings
- **regexp** Computes the percentage of the values matching a specified regular expression
- **grammar** Computes the percentage of the values that matched a grammar

The data-profiling module examines the rules file, collects all column-matching functions, accumulates their results when traversing the data source and finally includes them into the profile. The reasoning engine may then proceed to evaluate rules using backwards chaining and deduce whether a certain data element had a semantic data type. To illustrate this process with a simplified example consider the following rule:

```
(lookup(”country”)>0.6) OR (substr(name,”Country”)) AND (lookup(”country”)>0.3) =>
                        data_type|=”CNTR”
```

This rule relies on a lookup table of possible countries. It attempts to find the fields producing more than 60 percent of matches with the lookup table or alternatively the columns whose field name contains a indicative sub-string and at the same time where there is more than 30 percent of matches with the lookup table. By examining this rule, the data-profiling module will determine that it needs to perform the lookups into the countries table and to include the result into the profile. The reasoning engine, when asked to find a value of **`data_type`** variable, will thus be able to evaluate all the rules that potentially determine the data type (including the given one). If the condition of this rule was satisfied, the assignment will proceed and the identified data type will be stored as a value of **`data_type`** variable.

Our reasoning engine implements exclusive and inclusive assignment operations. With the help of the latter it is possible for variables to take multiple values. Due to this provision, we can build knowledge bases capable of expressing the data type at different levels of detail. For instance we can represent a situation where a certain field is identified as a home phone number or a fax number at a lower level of detail and simply as a phone number at a higher level. At the same time, with this mechanism, it becomes possible to find some data quality problems that are caused by evolving meta-data when values of several incompatible data types were found for the same field.

The reasoning engine is also capable of detecting inconsistencies in the knowledge bases such as contradictory exclusive assignments and circular reasoning loops. These capabilities are necessary for manual development of the rules.

Thus, at the end of the data type recognition process, we can associate existing data quality rules with all data elements for which a data type was identified. These rules are placed into the resulting scripting file and presented to the user for validation.

DISCOVERING NEW PROBABLE DATA QUALITY RULES

In some situations it is possible to discover a probable data quality rule by examining the data set and finding typical patterns that have good statistical support. We can assume that those patterns represent data quality rules and that the data elements that are not part of any pattern represent a probable data quality error.

We employ a variety of data mining algorithms to attempt data quality rule discovery. The applicability of a particular algorithm is decided based on the results of data profiling so that only the algorithms with a good chance of success are used to analyze a particular data element. There are several properties that we require from a data-mining algorithm:

- The algorithm must be able to tolerate a moderate number of errors in its training data sample
- The algorithm must estimate relevance of discovered rules
- The algorithm must have a reasonable computational complexity

Of the algorithms that we employ, some are relatively straightforward. For instance we perform one-dimensional clustering on numerical columns and compute distribution histograms for columns with a limited number of different values. If an unambiguous set of clusters is found for a numerical column, we can derive a data quality rule as a disjunction of numerical ranges that we expect the data to fall into. In the case of a column with a limited number of values, we discard the values with low probabilities of occurrence and derive the data quality rule as a lookup statement into the table of remaining values (i.e. those with high probability of occurrence).

We further briefly overview several other kinds of data quality rules and data mining algorithms that we found effective in discovering the rules.

Associative rules

There might be a dependency between particular values in different fields of a table. For instance, in a database storing city names and area codes we can discover a dependency such that every city has a limited set of allowed area codes. Through the use of data mining techniques such rules can be identified.

An associative rule can be discovered with the help of clustering techniques. By finding clusters of related

values in the multi-dimensional space of column attributes we can then produce logical descriptions that represent the clusters.

We use a number of different clustering algorithms. The selection of a particular one depends on the types of data in the training sample. For instance, when most dimensions contain values such that it is difficult to establish a similarity between them, we will use a binning technique to map every row from the data sample to an appropriate bin depending on the composition of attribute values. The bins with large numbers of items mapped are then identified as clusters.

When dealing with mostly numerical attributes we apply an agglomerative clustering technique using a number of various distance measures.

Both algorithms are tolerant to the presence of errors in the data sample and permit data samples of a considerable size. By examining sizes of discovered clusters we can also identify situations when the algorithm failed to find a clear set of patterns.

The discovered clusters are translated into logical descriptions and presented to the user as data quality rules.

In our experience, the relevance of discovered rules for the purposes of data quality was the best in the situations where only discrete value columns were associated or in the situations where a single numerical column was associated to one or several discrete value columns. Although this fact may well be due to imperfections in the discovery algorithms or the nature of the test data that was used, most purely numerical associations appeared incidental and uninteresting from the data quality perspective.

Arithmetic equation rules

Some data sets may not cluster well with the help of techniques discovering associations. However, there still may be a discoverable pattern in that case. One situation when the discovery is possible involves a numerical field that is arithmetically computed based on the values in other fields. In these situations it might be possible to discover an equation pattern.

A number of different methods were developed in the field of equation discovery. Early systems relied on multiple heuristics to discover regularities and trends in sample data sets [3]. We have examined two general techniques. Genetic programming methods use a large population of partial functions and recombine their elements in a locally optimal search for a best equation. An alternative brute force technique has been also examined where the search space is explored in a regular way. The search is defined by the equation grammar that is expanded to obtain a new candidate equation that must be verified against the data sample to check if the solution has been found. We came to a conclusion that the latter technique is more effective in most situations.

Presence of constants and coefficients in the equations expands the search space considerably making brute force solution computationally unfeasible for even relatively short equations. We use a very limited set of candidate constants and coefficients and implement several techniques to narrow the search space. The current implementation of equation discovery guarantees finding short equation (those using 3-4 binary functions). In some cases, it may also find longer equations.

The method of candidate equation verification allows for the presence of errors in the sample set. Given appropriate settings, the algorithm can also locate several equations for the same data set (such a discovery may signal an evolving data problem). The discovered equations are represented as data quality rules and passed to the user for validation and editing.

With the help of the equation discovery method we were often able to find many simple kinds of equations frequently appearing in business databases. Most of the time the discovered equations were

meaningful representing the true feature of the data. However since the elements of equations were almost always automatically calculated in the analyzed databases, very few data quality errors were found when checking the automatically discovered equations against the databases.

Regular expression rules

A regular expression is a computational device used to represent common patterns in sets of related text strings. These can be used as data quality rules in the situations where entries of some column contain short text words with common syntax.

A regular expression over some alphabet (e.g. *{a,b,c}*) is either an atomic expression consisting of a single symbol from the alphabet (e.g. *a* or *b*), or a sequence $r_1 r_2$ of any two regular expressions r_1 and r_2, or a disjunction $r_1 \mid r_2$ of any two regular expressions or, finally, if *r* is a regular expression then so is r^*. In the latter case the symbols produced by *r* are repeated *0* to *n* times. Additionally, brackets are used in a conventional fashion to disambiguate operator precedence. There are other syntactic additions that are possible, for instance, character classes (e.g. **`[a-z]`** to specify any symbol between **`a`** and **`z`**) [1], [2].

The following sample regular expression approximately describes the syntax of Canadian postal codes:

`[a-zA-Z][0-9][a-zA-Z]-[0-9][a-zA-Z][0-9]`

To attempt automated regular expression discovery we select a training data sample of a small size (100-200 words), build an over-specific automaton representing the entire data sample and perform a series of locally optimal merging steps to generalize the automaton [4], [6], [8].

Figure 3 illustrates this process for a trivial data sample consisting of only two strings: *{abb, abab}*.

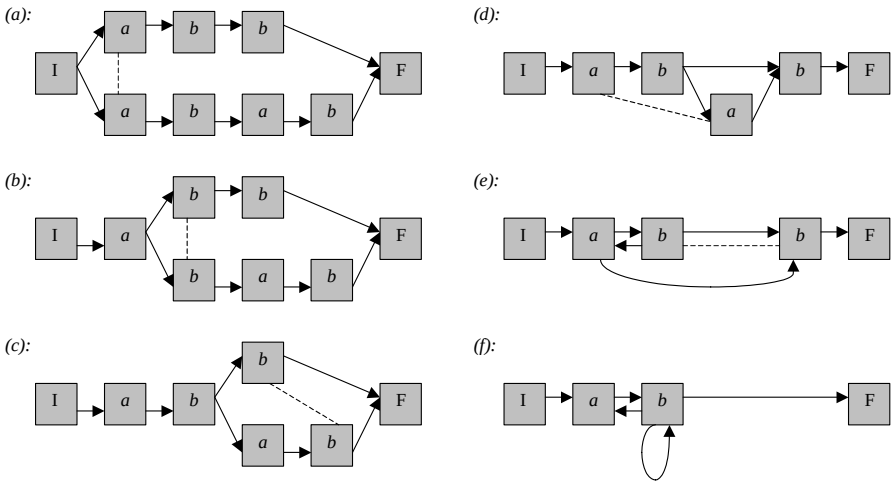

Figure 3: *Deriving regular expression automaton*

Once the automaton is derived, an extraction procedure is applied to recover the regular expression. In this case, the expression $ab^+(ab^+)^*$ will be recovered. Note that this expression generalizes the strings in the data sample and describes many other similar strings.

Multiple heuristics improving the practical performance of the algorithm were found. In particular we developed a special procedure to derive character classes and made overall improvements based on the

properties of separator characters.

The algorithm that we use is capable of discarding multiple strings from the data sample that are dissimilar from all other strings and are, thus, potential errors. If too many strings were discarded, we can assume that the data sample does not have a recoverable regular expression pattern and terminate the process.

The discovered regular expressions are recorded in the form of data quality rules and are presented to the user for validation.

In our experiments, with the help of this algorithm we were often able to discover simple syntactic patterns in test databases. Despite the algorithm's limitation on the size of its training data sample, quite often a correct regular expression was discovered even in situations where the sample was only a tiny fraction of the entire data set. In several situations (such as in the cases of regular expressions for postal codes or email addresses) the discovery was preempted by the identification procedure that had correctly identified the data element and located an already existing pattern for it. In other test cases, for instance those representing part numbers or document names, the discovery algorithm was able to find a meaningful pattern despite the presence of errors in selected training samples.

Grammar rules

Regular expressions describe a particular class of string sets. They seem to work well when the strings in the set are on average one short word in length. In the multi-word situations we, instead, attempt to discover a context free grammar.

A context free grammar over an alphabet (e.g. *{a,b,c}*), is a set of production rules of the form $X \rightarrow p_1^1 p_2^1 \ldots p_i^1 \mid p_1^2 p_2^2 \ldots p_j^2 \mid \ldots \mid p_1^n p_2^n \ldots p_k^n$ where X is a non-terminal symbol being produced and p_n^m is either a non terminal symbol (for which a separate production rule must exist) or a terminal symbol (in which case it should belong to the grammar's alphabet). Every sequence of productions has an equal probability of occurrence and every rule may consist of several sequences.

For example, consider the following context free grammar over the alphabet *{a,b}*:

$$G \rightarrow ab \mid aGb$$

This grammar has only one production rule and it describes an infinite family of strings *{ab, aabb, aaabbb, …}*. In general, unlike in this simple example, we can construct grammars where the alphabets consist of words as opposed to individual characters.

Context free grammars have a higher power of expressiveness compared to regular expressions. For instance, it is theoretically impossible to describe the sample string family presented above with the help of regular expressions.

The grammar discovery algorithm that we employ starts by representing every sample string as a production rule and performs a series of collapsing and merging operations [5], [6]. Consider an example of grammar discovery presented in **Figure 4**. As the figure demonstrates, the procedure executes a sequence of collapsing steps where a production rule is simplified by creating a new production rule followed by a merging steps that could unify several productions.

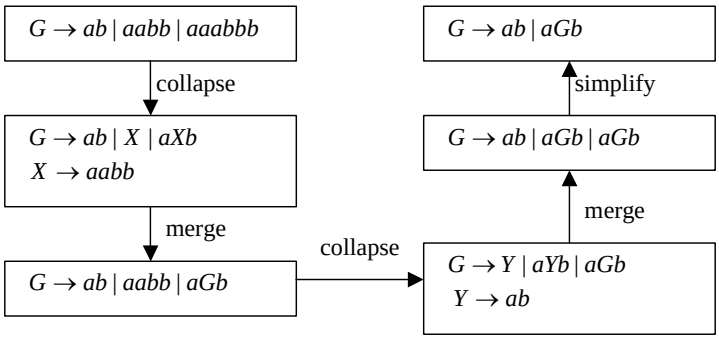

Figure 4: *Deriving context free grammar*

We employ a number of heuristics improving the practical performance of the algorithm that can also detect moderate number of errors in its data sample.

As in all previous cases, discovered grammars are presented as data quality rules for user's validation.

The context free grammar discovery algorithm that was implemented showed some promise in the cases of multi word textual columns where regular expression discovery was not feasible. However, the generalization capabilities of the current implementation appeared insufficient to produce a meaningful rule in majority of instances. We hope to be able to improve the effectiveness of this method by employing a series of preprocessing steps where similar individual words are identified.

CONCLUSION

The described automated information quality assessment software tool can be employed in a wide variety of settings from consultants assessing a possible bid, organizations evaluating the need for data quality improvement to analysts already working to resolve finer details of an ongoing project.

The created tool is quite flexible and allows expressing a wide variety of different data quality rules in a compact syntax. Tool's elements are extendible, for instance the user can improve the data type recognition knowledge base to increase tool's performance in the future.

There are multiple extensions and additions possible to improve the tool:

- Implementing the framework to handle database context rules
- Adding new data mining algorithms
- Improving the functionality allowing multiple data mining algorithms to work in concert
- Simplifying the scripting language
- Improving visualization module

Although at this early stage in development we have not yet gained a sufficient data validating system's performance on practice, the results of performed tests were often encouraging.

REFERENCES

[1] Angluin, D., and Smith, C. Inductive Inference: Theory and Methods. Computing Surveys. 1983, 3(15).

[2] Carrasco, R., and Oncina, J., Learning Stochastic Regular Grammars by Means of State Merging Method. *Proceedings Second International Colloquium ICGI*. 1994.

[3] Langley, P., Zytkow, J., Data-driven approach to empirical discovery. *Artificial Intelligence*, Vol. 40. 1989.

[4] Savchenko, S., Practical Regular Expression Mining and its Information Quality Application. *Proceedings Seventh International Conference on Information Quality ICIQ'02*. 2002.

[5] Stolcke, A., and Omohundro, S., Inducing Probabilistic Grammars by Bayesian Model Merging. *Proceedings Second International Colloquium ICGI*. 1994.

[6] Stolcke, A., Bayesian Learning of Probabilistic Language Models. *PhD thesis UC Berkeley*. 1994.

[7] Feldens, M., Lima, A., Lima L., Data Quality In Action: Challenge in an Insurance Company. *Proceedings Seventh International Conference on Information Quality ICIQ'02*. 2002.

[8] Goan, T., Benson, N., and Etzioni, O., A Grammar Inference Algorithm for the World Wide Web. *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*. 1996.

[9] Guimares., T., Staples, D., McKeen J., Empirically Testing Some Main User Related Factors for Systems Development Quality. *Proceedings Seventh International Conference on Information Quality ICIQ'02*. 2002.

[10] Wang, R., Product Perspective on Data Quality Management. *Communications of the ACM*, Vol. 41. No 2, 1998.