Generating rule suggestions in PBE data transformation

Kuldeep Reddy
Southeast University
Nanjing, China
brkuldeep@gmail.com

Abstract

This paper presents an idea for generating rule suggestions in performing programming-by-example data transformation operations. Data transformation is an important operation in data integration that is carried out frequently in many application domains, its programming-by-example variant consists of a grammar from which a rule is built. It is then used in creating a new data item from an existing data item by searching for occurrence of pattern data item and replacing it with a new data item defined in the rule. This paper proposes two new components in this system a) generating suggestions for new rules based on rules which have been given as input, in particular differences between structures in rule sequences b) suggesting rule modifications that would include an item user is interested in, with the final modified ruleset as a way to provide explanation to user why-not question.

1 Introduction

Data integration Data integration involves combining data residing in different sources and providing users with a unified view of them [1]. Various components of typical data integration process include - data quality assessment, record matching, schema matching, data transformation, data provenance assessment, data warehousing etc.

Database usability primarily concerns with developing techniques to make traditional databases more accessible to the users [3]. Examples of visual interfaces include faceted search, template based search, browsing query history and results. There has been recent work on developing query specification techniques involving just gestures and voice. Text based interface techniques include keyword search and natural language search. Miscellaneous techniques include spreadsheet based interfaces, query-by-example technique, techniques to handle the empty-result problem with query relaxation or reformulation etc, personalization and diversification techniques.

Data transformations Data transformations are often required to address the problem of heterogeneity in various data sources by converting the data from multiple sources into the same format. One common approach to handle data transformation is to define a transformation language and then generate rules based on the language to perform data transformation on a large set of data. However, this approach usually requires expert users to write individual transformations for each data source manually. To alleviate this problem, recently approaches have developed as in [2] that take user examples as input and build data transformation rules around it taking into consideration a grammar.

Query Suggestions An approach to solve the ambiguity and inaccuracy in the information retrieval system is query suggestion [4]. It is very common for a user to reformulate their query when they didnt receive ideal result from their original query. The system can improve the users searching effort by providing suggestions by guessing the user intention, according to either users past behaviour or data-driven techniques.

Why-not There have been many techniques proposed in literature that can explain where a piece of data comes from, this can explain surprises in a result set. The idea of whynot [5], on the other hand, seeks to explain why a certain item is missing from the result set. In the setting of traditional relational databases where the idea of why-not originated, the user provides the SQL query and a missing result item, the system then pinpoints where the item was not included in the resultset.

2 System design

The overall system prototype design involves two new components - a) a module to generate rule suggestions from the graph structure constructed to manage rule sequences c) a module to answer user why-not questions on data transformations and providing explanation in the form of modified graph structure. They key modules are described in more detail below.

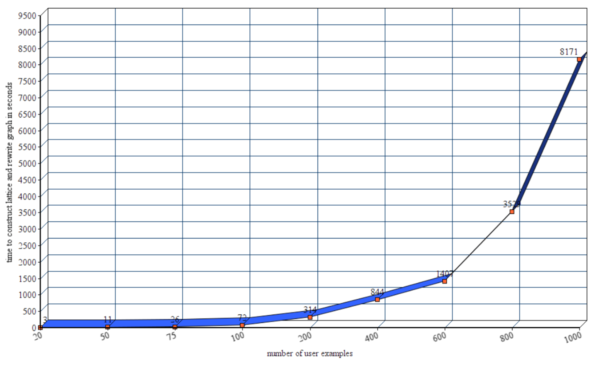
2.1 Generate rule suggestions

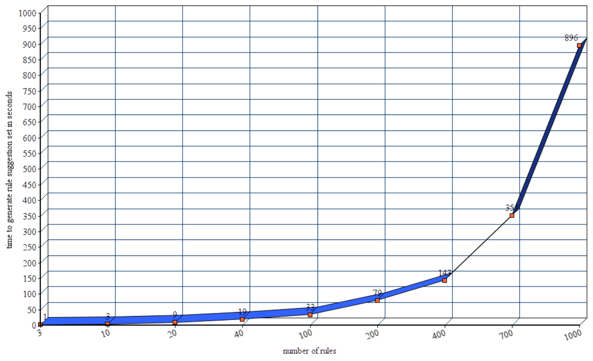
In the first part of system, the paper proposes a graph structure to store PBE data transformation rules. The rules we consider have been developed as part of foofah system [6], where the focus has been on improving its usability. By having a graph structure we are storing relationships between common components in rule sequences and if the new rule structure is similar to various combination of rules in the graph, it can retrieve relevant common rules more efficiently from graph instead of the inefficient repeated sequential searches. The rule sequences are organized in a graph structure described as follows. That is the rule sequence at the top of graph is represented as root and can be thought of as a union of all the rules. The rule sequence in the next level are the identifiers for individual rule sequences, which means if we have to build such a structure for 10 rules there will be 10 nodes with identifiers rule1, rule2, rule3 etc for each of the rules in level of the graph structure. The next level of the structure contain the individual rule components as part of a directed subgraph, which means for instance if the PBE data transformation program contains subrule sequences of the form split(t,1) and replace(t,2) then the level 3 of the structure will contain nodes denoted by these subrules in the form of directed subgraph with edges between them denoting the sequence of their execution. The next level contain nodes identifying the individual variables that would eventually point to say column names in a database or constant values or literals. The bottom level of the structure contain nodes for the various literal or numeric constant values. So, overall we end up with a structure consisting of 5 levels which can be used to manage the rule sequences, at the first level we have a root node, at the second level we have identifiers for individual rule sequences, at the third level we have nodes for subsequences in rule sequences with directions between them denoted their flow of execution, at the fourth level we have variable names and fifth level we have literal or numeric constant values.

In order to alleviate some of burden of writing all the rules for the user, this section proposes a way to generate rule suggestions using the graph structure designed above. This is an entirely data-driven approach that just makes use of rules written till now and does not require any additional data sources and only utilizes the graph structure developed earlier. The basic intuition behind the proposed approach is that whenever the user starts with a partial rule, it searches for the occurrences of the partial rules in the rules graph. Performing such a such on the rules graph is done by string similarity matching utilizing various measures in literature such as jaro-winkler, sorenson-dice etc. The suggestions are ranked according to the popularity score, which represents the frequency of the elements of new partial rule found in the earlier rules graph. Of course, utilizing more sophisticated string similarity heuristics is important and impact on the quality of suggestions generated and studying them is part of future work.

2.2 Answering why-not questions

In the second part of the paper, it proposes applying why-not questions in PBE data transformation. Frequently situations arise where the user is interested in knowing why a particular item is missing from the resultset. This concept has been developed extensively in relational query systems and of late in graph matching and clustering. This paper is the applies concept of why-not in this context of graph-based structure PBE data transfomation which is used to generate suggestions. The user specifies a data item that should have appeared in the transformed dataset but instead is missing from the transformed dataset. To know why the required dataset is missing, it first identifies the rule which generated either the data item itself exactly or the rule sequence that could have produced a larger transformation of in which the data item in question is part. Infact, there can be more than one rule that produced the data item in question. All the rule sequences which produced the data item in question exactly and those that produced larger transformed dataset of which it is part of are identified and which are then modified(through either insert, delete or move) to reflect the changes required by the user. Identifying such rules and modifications are done through string similarity algorithms and simulated annealing techniques. Designing better heuristics is part of future work. On the other hand, replacing the data item in question can affect other rules which have the data item relevant rule in question either exactly or as part of larger ruleset. Therefore, another round of rule identification is required to find rule sequences which have the rules with the data item in question. This again requires string similarity algorithms, of course studying better heuristics is again part of future work.





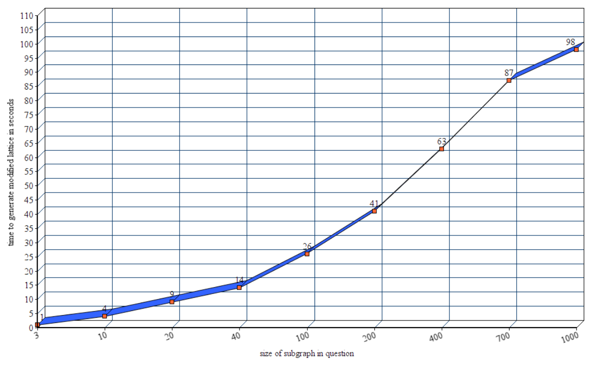
3 Experiments

The experiments were conducted on a 64-bit computer running windows 7 in Java. The real-world data used in experiments are obtained from the University of Florida Sparse Matrix Collection (cise.ufl.edu/research) and the Parasol project and KONECT(http://konect.uni-koblenz.de/networks).

The graph shown below in the figure 1 show the execution times in seconds to construct graph in PBE data transformation for various parameter configurations for sizes of rule 3 to 1000.

The graph shown below in the figure 2 show the execution times in seconds for the rule suggestion algorithm in PBE data transformation for various parameter configurations for sizes of graph that is the number of rules varying from 3 to 1000.

The graph shown below in the figure 3 show the execution times in seconds for the why-not algorithm in PBE data transformation for various parameter configurations for sizes of data item in question varying between 3 to 1000.



4 Future work

The purpose of the paper is to introduces a new approach to generate PBE data transformation rule suggestions and debug it. Designing more comprehensive complex solutions to the aforementioned problems is part of future work.

References

[1] Halevy, Alon Y. Data Integration: A Status Report. , University of Washington , Seattle, Washington, USA (2003)

[2] Wu, Bo, Szekely, Pedro A. and Knoblock, Craig A. "Learning Transformation Rules by Examples." Paper presented at the meeting of the AAAI, 2012

[3] Jagadish, H. V., Chapman, Adriane, Elkiss, Aaron, Jayapandian, Magesh, Li, Yunyao, Nandi, Arnab and Yu, Cong. "Making database systems usable.." Paper presented at the meeting of the SIGMOD Conference, 2007.

[4] Niu, Xi and Kelly, Diane. "The use of query suggestions during information search.." Inf. Process. Manage. 50 , no. 1 (2014): 218-234.

[5] Islam, Md. Saiful, Liu, Chengfei and Li, Jianxin. "Efficient answering of why-not questions in similar graph matching." Paper presented at the meeting of the ICDE, 2016.

[6] Zhongjun Jin, Michael R. Anderson, Michael J. Cafarella, H. V. Jagadish: Foofah: A Programming-By-Example System for Synthesizing Data Transformation Programs. SIGMOD Conference 2017: 1607-1610