Or2yw: OpenRefine Recipe to Yesworkflow Diagram*

First Author[1][0000−1111−2222−3333], Second Author[2,3][1111−2222−3333−4444], and Third Author[3][2222−−3333−4444−5555]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{abc,lncs}@uni-heidelberg.de

Abstract. OpenRefine is a powerful toolkit for data cleaning, where it saves data cleaning recipe in JSON format to promise reusability. While JSON files are both human and machine-readable, they are hard to comprehend in their entirety. We have constructed a user interface that transforms an OpenRefine recipe into a workflow diagram to make the protocol more transparent and understandable. OpenRefine recipes are ostensibly linear, but there are two levels of operations in OpenRefine, at 1) intra-column and 2) inter-columns. Dependency relationships in data cleaning workflow will change when activities occur at inter-columns. Besides, it is hard to follow lengthy and redundant recipe processes in the recipe. A high-level abstraction is needed to simplify those actions.
The or2yw toolkit addresses these problems by automatically transforming the original recipe in OpenRefine to a structuralized YesWorkflow model. YesWorkflow is a language-independent tool that can visualize a workflow based on particular annotation and can be used to recover the script's information. The workflow model following transformation is more transparent than the JSON recipe. Also, it can provide implicit operation provenance information at both intra-column and inter-columns in OpenRefine.

Keywords: OpenRefine · Yesworkflow · Provenance · dependency relationship.

1 Introduction

OpenRefine is a popular data cleaning toolkit that allows users to do data transformations in a browser-based, spreadsheet-like GUI [2]. Particularly, OpenRefine has a restricted interface to operate data cleaning workflow, combining the operations used in the transformation into a simple history list [1]. This operation history provides both prospective and retrospective provenance. For prospective

* Supported by organization x.

provenance, it collects the operations used in transformations and their arguments, i.e., prospective provenance needs to be captured before execution. Retrospective provenance is captured at runtime, such as the intermediate data produced [5].

However, operation history in OpenRefine is of limited transparency. For operations in OpenRefine, some can change both the table's structure and value (e.g., the operation *Add column based on this column* will create a new column and new values based on the original schema). The other type of operations will only affect the cell values (e.g., operation *To lowercase* will change cells' values in this column to fit for the pattern). Or2yw toolkit scales dependency on column level, where the former inter-columns transformations as geometry transformations, the latter intra-column ones as rigid transformations [6]. Geometry transformation will "break" the original table's data lineage, i.e., adding new dependencies or dropping old ones. Therefore, it is not easy to trace the lineage from a provenance perspective if there are breakpoints in the processes. Or2yw toolkit could help transpose operation history into the Yesworkflow diagram, helping return each operation's dependency relationships[4].

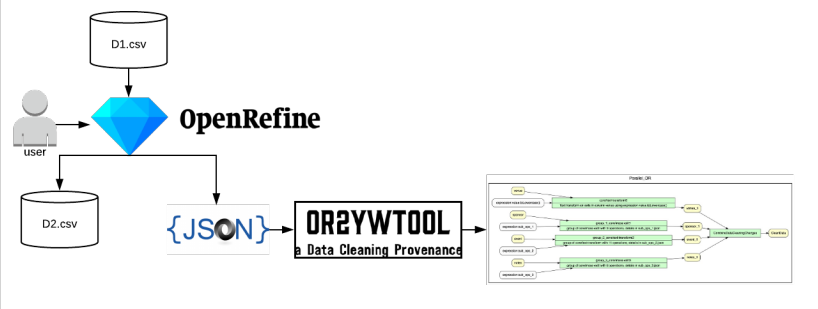

Fig. 1: a user cleaned a dirty dataset D1.csv file using OpenRefine. This process in OpenRefine will output a clean dataset D2.csv file and a data cleaning recipe in the JSON format which is not user-friendly. User then can transform this recipe using or2ywtool and generate a transparent workflow based on the recipe.

2 Transformation and Prototype

There are three modes in the or2yw toolkit, including sequential, parallel, and merge.

Given a transformation T_1 on column C_1, if the afterward transformation T_x also works on column C_1, then this following transformation is supposed to be dependent on the effect of T_1. In this situation, the structure of T_x and T_1 would be sequential. In or2yw, the sequential mode is by directly translate the

data cleaning workflow from the original JSON file (see Figure 2), step by step, without considering the dependency relationships among operations.

On the other side, if the transformation T_x applies to a different column, named column C_2. As the column C_2 is independent of column C_1, the T_x should be independent with the effect of T_1, and the structure of T_x and T_1 would be parallel. Here are certain conditions that define parallel operations.

Parallel Node: Operations in OpenRefine is applied row-wise and is stateless: no state is maintained among the processing of rows [1]. Thus, it is easy to parallelize the operations if the operations are executed under different columns, as they amount to a pure map on the list of rows. Operations on different columns are expected to be mutually exclusive and not affect each other, while any actions on the same column following those parallel operations will be executed sequentially.

Join Node: Two or more parallel operations will be joined together if there is one operation that combines two or more columns as the input parameter in the recipe. The joined operation will be executed respectively to the latest transaction on each column preceding this operation. E.g., operation *Add column based on this column...* allows users to concatenate multiple columns and create a new column.

Split Node: One or more new nodes will be generated if an operation produces multiple columns. The newly generated columns are mutually exclusive; therefore, they could be seen as a parallel node.

As defined in Data Transformation Algebra, Geometry transformation in OpenRefine would "break" the existing dependency relationships by creating new columns or deleting old columns [6]. To get the full provenance information on some column, especially the single cell's provenance, we need to make the dependency information in the data cleaning workflow explicit.

The merge mode mainly deals with a high volume recipe or has many "redundant" operations. Although the or2yw toolkit can produce the workflow, the output has so many nodes that it is too messy and complicated to trace. For this reason, we propose a method to merge those redundancies, where redundancy means the same operations with different parameters applying on the same column. By examining the column name, or2yw toolkit will find the same operations executed sequentially and join them together to produce only one operation node. A separated text file is provided, representing a sub-workflow containing complex operations and storing the parameter information as a node description. We can produce a typical workflow for an extensive process with redundant operations without losing the provenance with this approach.

3 Use case

We use dataset *menu.csv* from The New York Public Library, where it is a mix of simple bibliographic description of the menus (created by The New York Public Library) and the culinary and economic content of the menus themselves

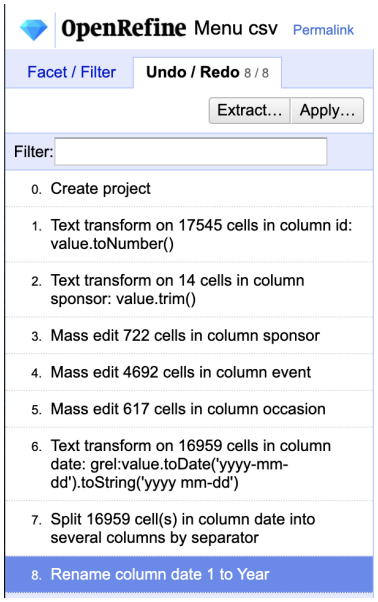

Fig. 2: Operation History in OpenRefine

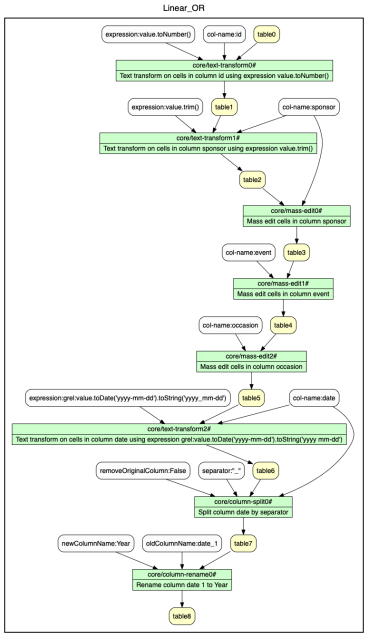

Fig. 3: Sequential: transpose the data cleaning into linear model.

(transcribed by users)[3]. We could expect how low the dataset's data quality could be that integrates manual and systematic input.

We apply eight transformations to the *menu.csv*, see Figure 2. Or2yw translates the recipe into a linear diagram, see Figure3. The green box represents transformation, the yellow box represents data stream in the data cleaning workflow, and the white box stands for the input parameters for each transformation.

Parallel mode is provided at Figure 4. Transformations on column *date*, *sponsor*, *occasion*, *event* and *id* are parallel nodes; Split node generates by applying column-split on *date*_1, where there are two new nodes generated, *date*_1_1 and *date*_2; In the final status, *CombineDataCleaningChanges* will merge all of the changed columns, where we could see this as Join node.

4 Conclusion

In this paper, we present the or2yw, a tool that can provide a more transparent provenance by bridging the OpenRefine and Yesworkflow system's data cleaning recipe. Or2yw also helps in extending the recipe by presenting the actual dependency relationships of the operations. Furthermore, the merge mode helps simplified the recipe and makes it more concise. This merge operation can produce an easy way to read workflow without losing detail in the provenance.

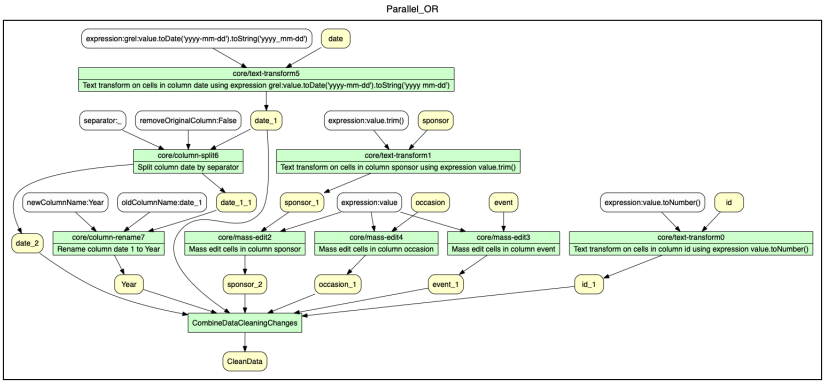

Fig. 4: Parallel: recover the real dependency relationships at inter-columns.

Our next step to provide better provenance for OpenRefine users is not limited to integrating this workflow tool with OpenRefine to give a workflow diagram directly from the application or web browser.

Finally, the or2yw is not the ultimate solution for providing provenance, but rather, an approach to make the Data Cleaning process more transparent.

References

1. Antonin Delpeuch. A complete language for faceted dataflow programs. *arXiv preprint arXiv:1906.05937*, 2019.
2. Lan Li, Bertram Ludäscher, and Qian Zhang. Towards more transparent, reproducible, and reusable data cleaning with openrefine. *iConference 2019 Proceedings*, 2019.
3. The New York Public Library. *Whats on the menu?*, 2020 (10/01/20).
4. Timothy McPhillips, Tianhong Song, Tyler Kolisnik, Steve Aulenbach, Khalid Belhajjame, Kyle Bocinsky, Yang Cao, Fernando Chirigati, Saumen Dey, Juliana Freire, et al. Yesworkflow: a user-oriented, language-independent tool for recovering workflow information from scripts. *arXiv preprint arXiv:1502.02403*, 2015.
5. Leonardo Murta, Vanessa Braganholo, Fernando Chirigati, David Koop, and Juliana Freire. noworkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop*, pages 71–83. Springer, 2014.
6. Santiago Núñez-Corrales, Lan Li, and Bertram Ludäscher. A first-principles algebraic approach to data transformations in data cleaning: Understanding provenance from the ground up. In *12th International Workshop on Theory and Practice of Provenance (TaPP 2020)*, 2020.