Towards More Reproducible Data Wrangling with OpenRefine*

Lan Li[1], Qian Zhang[2], and Bertram Ludäscher[1]

[1] University of Illinois at Urbana-Champaign, Champaign, IL, USA
{lanl2,ludaesch}@illinois.edu
[2] University of Waterloo, 200 University Ave W, Waterloo ON N2L 3G1, Canada
q394zhan@uwaterloo.ca

Abstract. OpenRefine (OR) is a popular data wrangling tool. During data cleaning , not only a processed dataset will be generated but also some other provenance-related byproducts. One of them is a native OR recipe (JSON file), and the other one is the Operation History (OH) that lists a series of human-readable data cleaning steps, both of which promote research transparency to some extent by containing some (but incomplete) prospective provenance and partial retrospective provenance information [1], making them difficult to be directly used for reuse and Reproducibility from OpenRefine Web API (see Fig. 1). In this poster, a prototype consisting of two sub-systems, one of which extends the native OR recipe to generate a complete recipe (a.k.a. enhanced receipt) followed by the second re-runner system, is created to complement the missing information between the actual data cleaning operations and the native OR recipe, which meanwhile facilitates transparency, reproducibility and reusability.

Keywords: OpenRefine · transparency · reproducibility · reusability

1 Introduction

OpenRefine is used for data wrangling and data cleaning in many areas nowadays. Data cleaning can be understood as taking (presumably "messy") input data $d1.csv$, subjecting it to a data cleaning procedure or workflow W, resulting in an (presumably "clean(er)") output data $d2.csv$[1] (see Fig. 1). When presenting $d2.csv$ three properties are often considered important: a). **Transparency**: Being transparent about what happened to $d1.csv$ to obtain $d2.csv$. For example, to document everything that happened, in particular make sure W'0s operations

* This work was supported by the US National Science Foundation NSF: OAC #1541450

[1] The desiderata for $d2.csv$ include that it is a better or more accurate model of reality than $d1.csv$, and that $d2.csv$ is fit for use/purpose (whereas $d2.csv$ might not have been). But this is not our concern here.

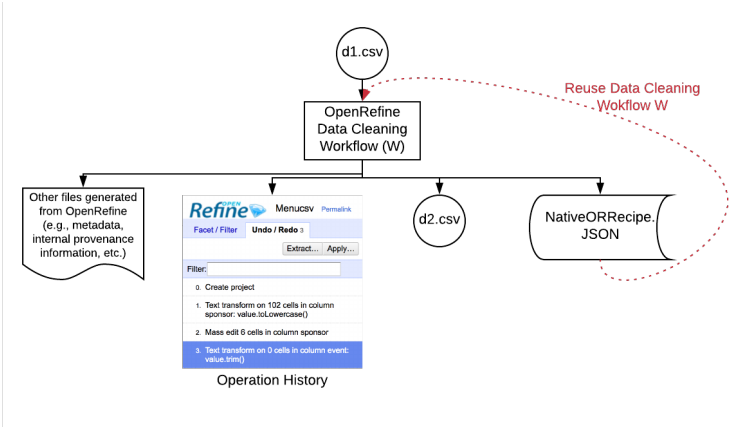

Fig. 1. Data Wrangling in OpenRefine

(and parameters) are clearly documented, which is primarily about retrospective provenance [2]. b). **Reproducibility**: Being closely related to transparency. The focus lies in that a user should be able to take the workflow W (possibly re-implemented in another language W), apply it to $d1.csv$ and obtain the same result, i.e., let $d3 = W'(P, d1)$, where P are any additional inputs (in particular parameters) that are needed to re-execute W on $d1$ to make sure we obtain the same[1] output $d2$. Then $d3 = d2$. And c). **Reusability**: The data cleaning workflow W developed originally for $d1$ should be readily applicable to similar dataset $e1.csv$. That is, we want to reuse W on $e1.csv$ and become a meaningfully better dataset $e2 = W(e1)$. The strength of OpenRefine lies in that it has already supported transparency, reproducibility and reusability to some extent, in a way that the operation history (a.k.a. recipe) provides a JSON representation (a.k.a. native OR recipe) of most of the operation that OR has performed on a given dataset/project. However, after studying OR recipes and other "sideways/internal provenance information" of OR, we found the following shortcomings: First of all, the native OR recipe shows limited transparency. For example, when the user performs some common transformations such as *To lowercase* operation, native OR recipe only captures the operation name **without** recording the actual change of the records (i.e., "from": [...], "to": ...). The other extreme example is the "single-cell edit" operation, for which neither the operation name nor the operation effect is recorded at all in the native OR recipe (see Fig. 2). This also demonstrates OR's limited reproducibility issue. Another problem with the native OR recipe is its limited reusability. For example, when the *Cluster and edit* operation is performed, OR captures only the actual change of each record **without** the underlying cluster method details such as the cluster

[1] In more general settings same could be relaxed to equivalent. Here, for simplicity, by same we mean the same bit-level representation (in practice: a diff between $d2$ and $d1$ will show no differences).

type, function and/or associated parameter settings. In this way, this native OR recipe can not be reused with the updated dataset.

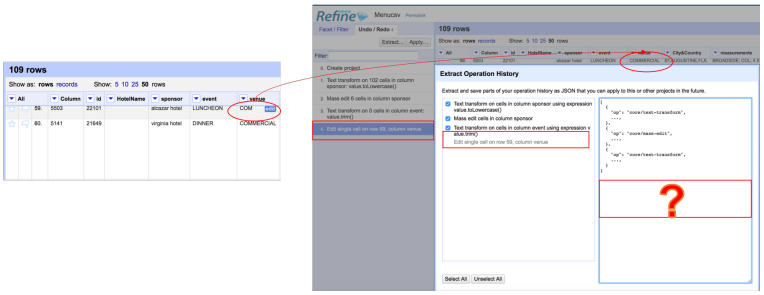

Fig. 2. Missing Info with Single Cell Edit Operation

2 Contributions

In this poster, a CLI (command line interface) prototype consisting of two subsystems is developed by extending the OpenRefine Python Client Library (OR-client) [3]. The first one is named CLOPER (Command-Line OpenRefine Prototype for Enhanced Recipe), which makes the incomplete native OR recipe complete to improve OR's transparency and reusability. And the ERRR system (Enhanced Recipe Re-Runner) verifies the resproducility of the enhanced recipe derived from CLOPER.

3 Prototype

The prototype includes two sub-systems.

3.1 Command-Line OpenRefine Prototype for Enhanced Recipe

CLOPER (see Fig. 3) aims to enhance transparency and reusability of the native OR recipe, which reads in the original "messy" dataset ($d1.csv$) and communicates with an OR server through the interface provided by the OR-client. The outputs consist of three products: an enhanced recipe ($EnhancedRecipe.JSON$) is generated at the back-end; a "cleaned" dataet ($d2.csv$) and a native OR recipe ($NativeORRecipe.JSON$) are exported from the OR web UI.

3.2 Enhanced Recipe Re-Runner

In regards to reproducibility, ERRR (see Fig. 4) re-implements the enhanced recipe ($EnhancedRecipe.JSON$) that is derived from CLOPER, applies to the

same original "messy" dataset ($d1.csv$). Again ERRR connects to an OR server via OR-client and obtains the same output ($d2.csv$) associated with the native OR recipe ($NativeORRecipe.JSON$).

A subset of the OR operations (e.g., *Cluster and edit*, *To lowercase*, *To date* , etc.) were used to test both sub-systems of the proposed prototype. The results showed that the enhanced OR recipe improved the transparency, reproducibility and reusability compared to the native OR recipe.

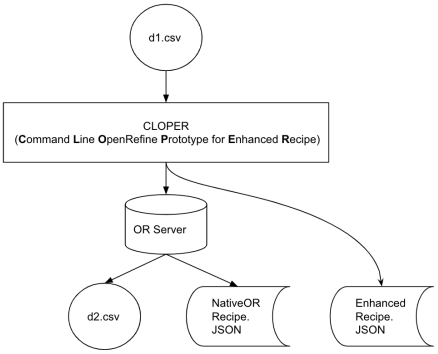

Fig. 3. Command Line OpenRefine Prototype for Enhanced Recipe(CLOPER)

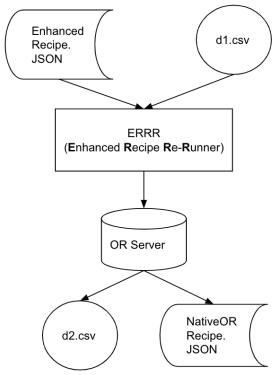

Fig. 4. Enhanced Recipe Re-Runner(ERRR)

4 Further Work

For the further work, more operations (e.g.,*single-cell edit*) need to be improved by fixing the incomplete and/or missing information (see Fig. 2). In order to advance reproducibility of the enhanced recipe, CLOPER is expected to assist in capturing more prospective and retrospective provenance.

References

1. Dey, S., Belhajjame, K., Koop, D., Raul, M.,Ludäscher, B.:Linking prospective and retrospective provenance in scripts. In: Theory and Practice of Provenance. In: 9 International Journal of Digital Curation. (2015)
2. Zhao, Y., Wilde, M., Foster, I.:Applying the virtual data provenance model. In: International Provenance and Annotation Workshop pp. 148-161. Springer, Berlin, Heidelberg.(2006, May)
3. OpenRefine-client Library Github, https://github.com/opencultureconsulting/openrefine-client.