In Search of Useful Collection Metadata: Using OpenRefine to Create Accurate, Complete, and Clean Title-level Collection Information

By: Kate M. Hill

Hill, K. (2016) In search of useful collection metadata: Using OpenRefine to create accurate, complete and clean title-level collection information. *Serials Review, 42*(3), 222-228. doi: 10.1080/00987913.2016.1214529

Abstract:

University of North Carolina at Greensboro (UNCG), like many libraries, recently migrated to a new knowledgebase and integrated library system (ILS) and found they had to clean up a great deal of messy serial title list data. In their search for solutions, they discovered the free, open source tool OpenRefine, a software program specifically designed for data normalization, transformation, and cleaning. This article describes the steps that UNCG used to take a publisher's title list file and transform it into a file format usable by their ILS. In doing so, this article will discuss major types of functionality in OpenRefine: downloading the software, importing data correctly, using the interface, transforming data on a column and cell level, exploring and normalizing data, and exporting files out of OpenRefine. At the end of this article, the readers should understand how to use OpenRefine on a basic level and be able to begin to use it on their own data.

Keywords: data software | electronic resource management | knowledgebase | OpenRefine, serials data | serials management | serial title lists

Article:

Introduction

Frequently, electronic resources and serials librarians find themselves inundated by messy publisher and knowledgebase data that they must edit and massage so it can be used by existing library systems. While prepopulated knowledgebases supporting discovery services and next-generation integrated library systems (ILS)—like Serials Solutions, OCLC Worldshare Collection Manager (WMS), and EBSCO Discovery Service—have relieved some of the pressure on serialists, issues remain. For example, at the University of North Carolina at Greensboro (UNCG), newly purchased collections often are found to be missing from our current knowledgebase and have to be manually added. Due to this and other issues arising from a recent ILS migration, we were in need of methods that could make data cleanup a simpler and

less time-consuming process. One of the most useful products we discovered is a free tool called OpenRefine (www.openrefine.org). This piece of open source software takes digital textual data that exists in a relatively complete state and, through a simple graphical user interface, lets the user quickly remove duplicate information, unify naming conventions, and transform data from one format (e.g., publisher-provided title lists) into another (e.g., a format required for upload into a knowledgebase or catalog.) This article, based on a presentation given at the North Carolina Serials Conference, will present a high-level overview of how one library used OpenRefine to manage and fix continuing resource data in their knowledgebase and catalog.

Background

OpenRefine, originally called Google Refine, is a community-maintained open source software project. Once downloaded, it runs locally on a desktop and does not require an Internet connection. However, its interface, a tabular editor that allows manipulation of data by columns and cells, does display via a web browser. This article will discuss in more detail some of OpenRefine's basic functionality, including methods to facet and explore data, how to perform mass transformations of data in a column, and tools to catch issues with data normalization. There are a great many more things that can be done with OpenRefine than what can be covered in this article. Of special interest to libraries is its ability to reconcile data with existing linked data sets—such as LCSH (Library of Congress Subject Headings), JournalTOCs (Journal Tables of Contents), and FAST (Faceted Application of Subject Terminology)—via strong application program interface (API) integration (Bedoya, 2014). Librarians who are comfortable with regular expressions and simple scripting can also develop much more complex data transformations using OpenRefine's specific regular expression scripting language, General Refine Expression Language (GREL). Even without programming knowledge, however, many common GREL expressions can be found online due to the wide adoption of this tool.

While this tool is relatively new to UNCG, it is not new to library projects. Jacquie Samples, head of Electronic Resource and Serials Cataloging at Duke University Libraries, presented at the 2014 Electronic Resources and Libraries Conference on her use of OpenRefine to do a mass cleanup of database catalog records before a migration to Serials Solutions (Samples, 2014). St. Olaf's special collections library used OpenRefine to discover issues with name and subject authority in their Nordic American Book Imprint Collection (Becknell & Weeks, 2013). Margaret Heller, digital services librarian at Loyola University Chicago, has used OpenRefine to take curriculum vitae (CV) data from faculty members and transform it into a format that could be brought into her Institutional Repository (Heller, 2013). Indeed, in a 2014 survey done of OpenRefine users, Magdmartin discovered that one out of four identified as librarians (Magdmartin, 2014).

My own use of OpenRefine began at North Carolina State University (NCSU) during my involvement with the Global Open Knowledge Base (GOKb), which is “an open data repository that contains key publication information about electronic resources as represented within the supply chain from publishers to suppliers to libraries” (About GOKb, 2016). NCSU's development of GOKb required gathering a large volume of publisher serials data and then transforming it into a modified form of Knowledge Base and Related Tools (KBART) for ingest.

Due to the extensive data cleanup and transformation needed, the development team decided to use OpenRefine as the basis of the data ingest system. When I went to UNCG, I quickly noticed that the implementation of OCLC WMS had left many collections either poorly represented or unrepresented and realized that applying the techniques used for GOKb would help solve data cleanup problems for serial collections at this new institution.

Installing and setting up OpenRefine

OpenRefine can be downloaded from openrefine.org/download for Windows, Mac, or Linux. This is a Java application, so users should have the most updated version of the Java Runtime environment installed. Once extracted from its zip file and installed wherever desired on one's computer, it will open in a browser window. While any browser should work, Firefox and Chrome tend to give the best results.

Starting a new project is as simple as selecting *Create Project* on the initial interface. For the projects discussed here, the most common types of data to import come from Excel files or tab-delimited files, though many other types of files, like Resource Description Framework (RDF) and Hypertext Markup Language (HTML), can also be used. Before importing any file, it should be noted that one of the limitations of OpenRefine is that it does not have as good an interface for data entry as something like Microsoft Access or Excel. Thus, a best practice is to have one's data set as complete as possible before importing it into OpenRefine. Once a file is imported, OpenRefine attempts to detect its type automatically and gives an appropriate preview of what the file will look like within the OpenRefine interface, as seen in Figure 1.

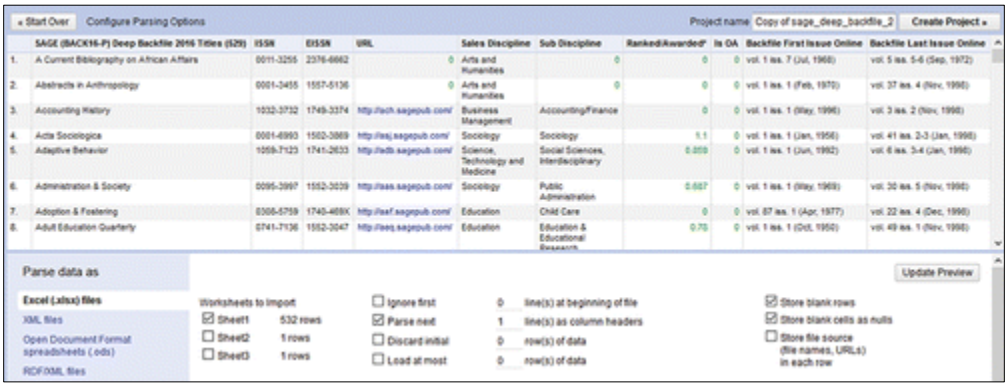


Figure 1. OpenRefine import interface.

When importing files from publishers, there are a few things that should be checked for in the preview. These include:

- Using *parse next X rows as column headers* to indicate which row should be used for column names;
- Using the *ignore first X lines at the beginning of the file* to get rid of unnecessary information;
- Parsing everything as text if bringing in tab-delimited or Excel files. Not doing this may result in conversion errors, especially with dates and numbers; and,

- Giving a file a name that is easy to remember. For projects at UNCG, the naming convention is <knowledgebase collection name>_<publisher name>_year.

Manipulating data at the column level

After importing a file and setting the correct import parameters, the next step is to make column-level changes. At UNCG, all files that will be ingested into our knowledgebase must be transformed into a modified form of the KBART standard, which includes all of the standard KBART fields and a few fields unique to OCLC. In addition, the fields must be in a specific order and no fields can be missing, even if they contain no data. Because of these strict formatting guidelines, most files we get from publishers need to go through some form of column-level change and reorganization.

OpenRefine lets one perform column manipulation quickly and easily. At UNCG, we first look at the current column names and determine what KBART field they correspond to, if any. Then we rename the existing columns to match the correct KBART field. OpenRefine uses column-level dropdown menus for most of its interaction, though there is an *All* column that allows for manipulation of the entire data set. To rename a column, one goes to that column's dropdown menu, then to the *edit column* submenu and selects *rename this column*. After this, we then locate the columns that contain information not corresponding to any KBART field. Using the far left column, the aforementioned *All* column, we select that column's dropdown, then pull up the *edit columns* submenu and select the *rearrange/remove* option. This brings up a drag-and-drop interface, as seen in Figure 2. This lets UNCG in one action remove all unnecessary columns and also rearrange all other columns into the order required by our knowledgebase.

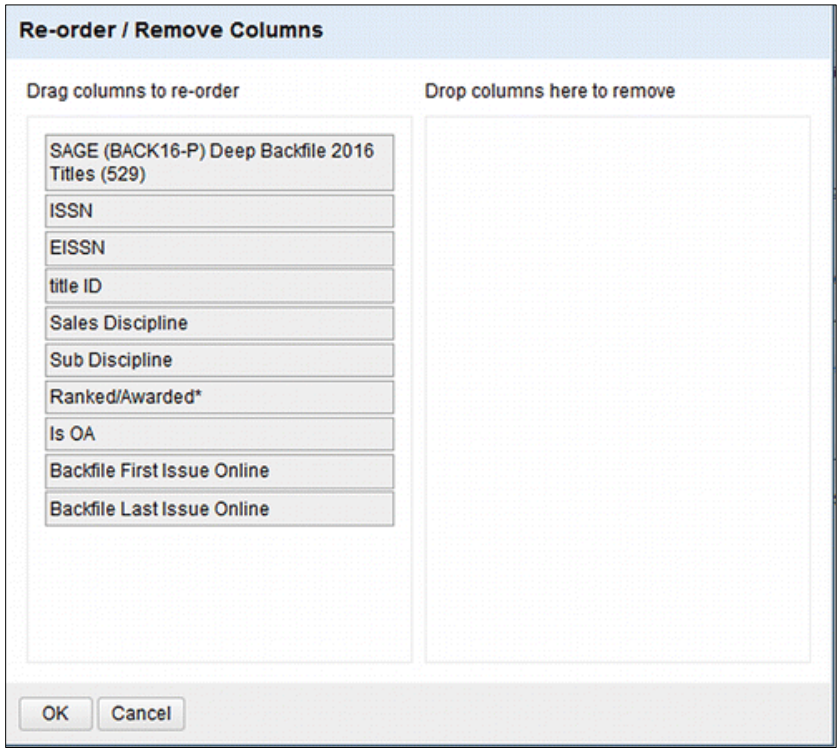


Figure 2. Re-order/Remove column interface.

Next, we begin to add columns to represent KBART elements not found in the publisher file. This is one of the places where OpenRefine does not work as well as Excel, as one has to go into a column's dropdown menu, click on *edit column*, and then select *add column based on this column*. This brings up another method of interacting with data in OpenRefine, which is the data transformation screen, as seen in Figure 3.

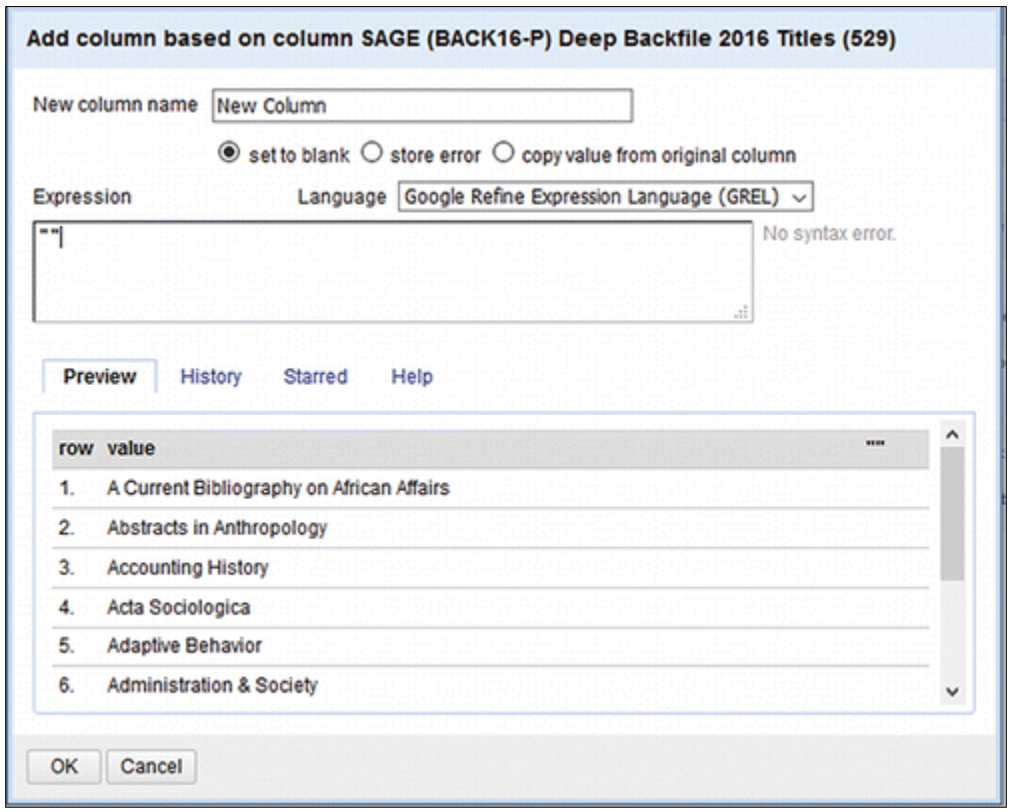


Figure 3. Adding a new blank column in OpenRefine.

Here one can give the column the appropriate name and also change what will display in every cell. The default is what had appeared in the column previously, but it can be edited to the expression "." This will create a blank cell, which can be seen in the preview box below the main expression interface. This preview box appears any time an expression is used to transform cell data and allows one to see if a transformation will work before it is applied.

Finally, we sometimes have data that are located in one column—generally date, volume, and issue data—that need to be in separate columns for KBART purposes. Luckily, OpenRefine makes this relatively simple. Using the *split into several columns* option under the *edit column* submenu, a new window appears that allows one to indicate what separator to use (generally a comma, period, or space) and how many times to perform this separation. Sometimes, this can require a bit of trial and error, and I often find that I have to do one split to separate volume, issue, and date and then another split to remove the volume and issue labels from their related numbers.

Exploring and normalizing data

Once the general file structure is correct, UNCG examines the data the publisher provided to discover what type of errors and inconsistencies exist. OpenRefine provides a few ways to do this. The first method is faceting data. The *facet* option appears under each column's dropdown menu. Once selected, it brings up a submenu, which allows one to facet by text, number, date, or a custom value. There are also prebuilt customized faceting methods, which can facet values by word length or blank cells. At UNCG, we generally go through each column and facet by text to identify blank cells and clear issues with formatting, such as groups of URLs that have no preceding http://. We sometimes facet by date, which presents the data as a timeline. This can help identify outlying publication dates that are likely incorrect. For information that should be a specific length, like International Standard Serials Number (ISSN) or OCLC number, faceting by word length displays a number line, which lets the user quickly see what cells do not fit the range of possibilities for the given ID. Once a facet has been selected, the information appears on the left panel of the interface, as seen in Figure 4.

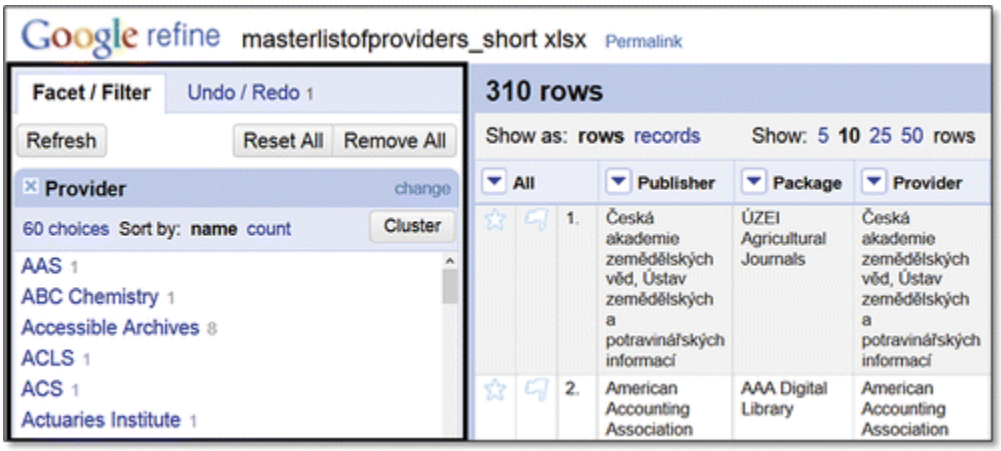
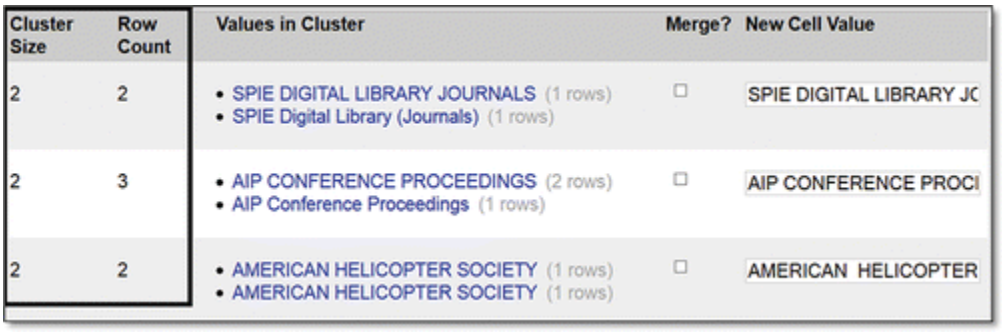


Figure 4. Displaying facets in OpenRefine.

After facet selection, we examine the values to see any patterns, trends, or issues appear. For text facets, one clicks on the value that is of interest and can choose to either *include* or *exclude*this value. Once a value has been included, the display changes to show only the rows whose cells contain the selected value. Building on what has already been selected, the user can add additional values from the same column or other columns. The ability to select and view multiple values across multiple columns lets the user create what is essentially a custom data query, minus the Structured Query Language (SQL) coding. All of this flexibility provides a rich method of exploring and limiting data. In addition, if there is clearly an error in a value displayed in the facet list, one can choose to edit that value. The edit made will affect all cells that contained the original value.

The other main way to examine and normalize data is through clustering. This technique is mostly useful in a field where duplicate values are expected. In a KBART file, this is mostly relevant to the publisher_name field. Clustering, which can be accessed via a column's *edit*

rows submenu or from within the *facet* panel, takes the user to a separate interface, seen in Figure 5.



Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	2	• SPIE DIGITAL LIBRARY JOURNALS (1 rows) • SPIE Digital Library (Journals) (1 rows)	☐	SPIE DIGITAL LIBRARY JC
2	3	• AIP CONFERENCE PROCEEDINGS (2 rows) • AIP Conference Proceedings (1 rows)	☐	AIP CONFERENCE PROCI
2	2	• AMERICAN HELICOPTER SOCIETY (1 rows) • AMERICAN HELICOPTER SOCIETY (1 rows)	☐	AMERICAN HELICOPTER

Figure 5. OpenRefine's clustering interface.

Here, OpenRefine provides a list of values that are similar enough that they might represent the same thing. A variety of algorithms are used to determine this similarity. This OpenRefine interface then lets the user decide which value is correct and, once selected, change all other values determined to be similar to this correct value. This allows the user essentially to develop simple authority control on the fly. At UNCG, I find the best practice is to start with the most conservative matching algorithm, called Key Collision, and move to the most lenient. More detail about how each algorithm works is available at https://github.com/OpenRefine/OpenRefine/wiki/Clustering.

Transformation

At UNCG, the final step is to fix, via cell editing and transformations, all of the issues discovered during the previous processes. In addition to the techniques already mentioned, such as splitting columns and editing via facets, OpenRefine has many other editing options. Clicking on any individual cell will bring up an edit window, where the user can change that individual value. Where OpenRefine really excels, however, is in the mass transformation of data on a column level. From the column dropdown, there is a submenu called *Common Transformations*. This submenu lists prebuilt transformations that can be applied without having to craft expressions. *Common Transformation*s affect every cell in a column. They include changing values to date, text, or number and changing text to all lowercase, uppercase, or titlecase. At UNCG, we regularly use the titlecase transformations on the publication_title field in our KBART files. For every column, UNCG also uses the common transformation *Trim Leading and Trailing Whitespace*, which removes all blank carriage returns around a string and *Collapse Consecutive Whitespace*, which then removes any double (or more) carriage returns. Since having extra spaces can quickly confuse faceting and also create unexpected results when splitting columns, performing these two transformations for every column makes sure other actions run smoothly.

After all of the common transformations necessary have been run, the final, and often most challenging, step is to create custom transformations. The custom transformation interface can be reached by going into the *edit cells* submenu and selecting *transform*. This interface acts in a

similar fashion to the interface discussed in the Manipulating Data on a Column Level section, with a few differences in the expression field, as seen in Figure 6.

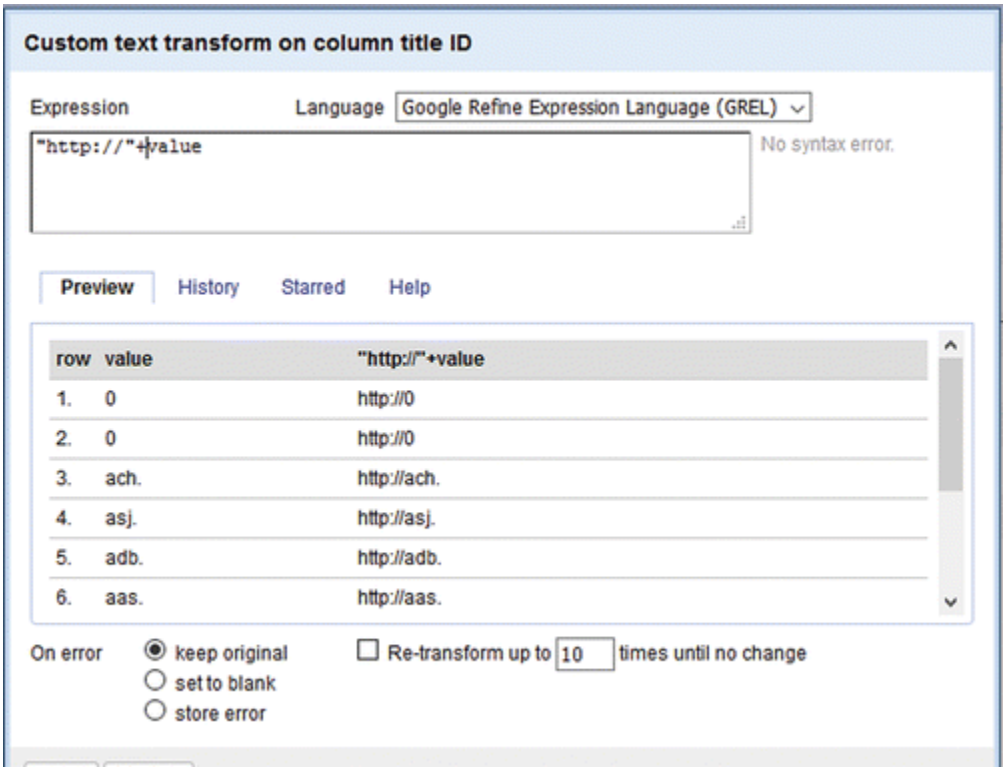


Figure 6. The custom transformation interface.

The default value here appears simply as value. This value represents whatever value is currently in a given cell. Thus, any modifications one makes to the value will affect all cells in a column. This also means that if the user wants to transform the value in any way and have exactly what they type appear, it must be written in quotes. The general way to manipulate a value is through GREL. Common commands that UNCG uses include using the plus sign to add material to either the front or end of every cell value. For example, if I have a group of journal codes but no title-level URLs, and I know that the journal codes form part of the URL, I can take the journal code and place the surrounding URL around it, like this: “http://“+value+”.bestpublisherever.com”. In OpenRefine, there are also a set group of commands called *functions* that can be used after the value to indicate that a certain action should be performed. The function itself is written as value.function. Whatever information, referred to as an argument, on which one wants the function to work is written between parentheses after naming the function. For my own work with publisher title lists, I use a few commands extensively. These include value.replace, which takes one text string and replaces it with another, and value.chomp, which removes one text string from the end of a cell value. Examples include:

Replacing & with and would be written as:

Value.replace (“&”, “and”)

Removing all trailing commas, periods and colons from the end of cell values is written as:

Value.chomp (“.”).chomp(“,”).chomp(“:”)

These are just a few examples of expressions that can be used in OpenRefine to do mass transformations. This part of this tool is one of the most powerful aspects as well as one of the most programming intensive. A more extensive list of functions can be found at https://github.com/OpenRefine/OpenRefine/wiki/GREL-Functions. For the type of manipulation done on publisher title lists, the string functions (the functions that work with text) are generally the most useful.

Another aspect of OpenRefine that is not directly tied to transformations, which I have found immensely useful while learning how to construct transformation expressions, is the *Undo/Redo* panel on the left side of the interface (see Figure 7).

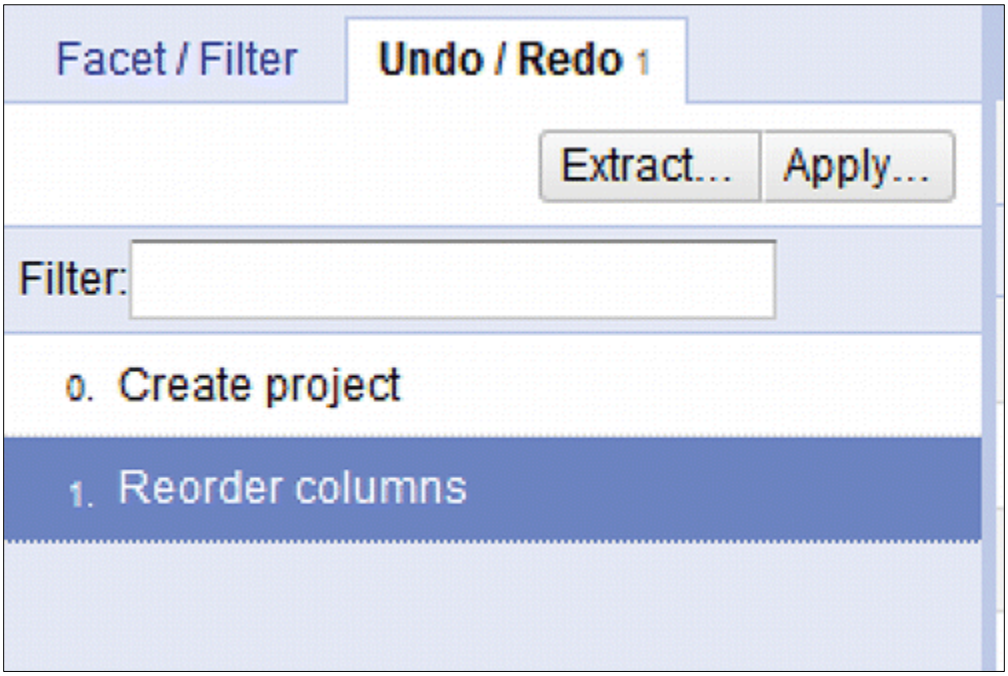


Figure 7. OpenRefine's undo/redo interface.

This panel lists every action that the user has taken in OpenRefine that has caused a change to the data set. Thus, faceting without editing is not included. By clicking on any of the actions, the user can reset the data to how it appeared before that action was taken. This ability allows novice users to experiment and try new things, safe in the knowledge that whatever they do can be undone. As such, OpenRefine is very beginner-scripter friendly and can serve as a practical tool for learning or improving one's knowledge of regular expressions and data manipulation.

Saving and exporting one's work

Once all transformations have been completed and the file exists in its final form, UNCG exports the file as a tab-separated value for ingest into OCLC WMS via the *export* button at the top of the right side of the interface. Though it may seem from this article that the entire process of working with a file takes place in one session, this is not the case. OpenRefine automatically

saves all files when the program is closed. When OpenRefine is restarted, files in progress can be returned to via the *Open File* option.

In addition to exporting a file after work is complete, UNCG also examines whether some of the steps taken to improve the current file could apply to other files by the same publisher. If we believe this to be the case, we select *extract* in the *Undo/Redo* panel to bring all the steps the particular file has gone through into a text file. One can choose to extract all steps or just some, though if using the latter option, it is important to pay attention to whether the steps chosen work without their full context. Once the steps have been chosen, UNCG copies the text file and saves it as <publisher name>_OpenRefine_macro. When working with a file by the same publisher, UNCG can then grab the corresponding text and, also within the *Undo/Redo*panel, select *apply*. This brings up a text box that we can then paste in the extract code. This will run through all the steps on the code that it can. This ability to create automatic macros can be a very powerful tool for creating a more efficient workflow.

Conclusion

Librarians, especially those dealing with electronic resources and serials management, frequently have to modify, transform, and fix messy title list data, be it from their own knowledgebase or a publisher file. This article describes OpenRefine, one tool that one library has found to be highly useful in its own work. Its powerful yet user-friendly interface; built-in space for experimentation; and ability for users to query, normalize, and perform mass transformations on data without programming skills make it an important product for librarians to know and understand. While it does have some issues, notably that it requires the use of regular expressions and a basic understanding of scripting and APIs to fully take advantage of all its functionality, most librarians, as demonstrated by the processes described in this article, do not need to engage with these areas in order to find OpenRefine useful. With this article and the further resources listed in the appendix as guides, the reader is encouraged to download OpenRefine, play, experiment, and see how it can be integrated into one's own library.

Appendix: Further resources

OpenRefine Official Documentation

Main Refine Page: http://openrefine.com

Official OpenRefine FAQ: https://github.com/OpenRefine/OpenRefine/wiki/FAQ

Screencasts introducing
OpenRefine: https://github.com/OpenRefine/OpenRefine/wiki/Screencasts

OpenRefine Wiki: https://github.com/OpenRefine/OpenRefine/wiki/

General tutorials

Free Your Metadata: http://freeyourmetadata.com

Getting Started with
OpenRefine: https://wikis.utexas.edu/pages/viewpage.action?pageId=46631837

A Librarian's Guide to OpenRefine: http://acrl.ala.org/techconnect/post/a-librarians-guide-to-openrefine

Using OpenRefine to Clean Multiple Documents in the Same Way: http://schoolofdata.org/2013/07/26/using-openrefine-to-clean-multiple-documents-in-the-same-way/

Chitchat with New Datasets: Facets in OpenRefine: https://blog.ouseful.info/2012/11/06/chit-chat-with-new-datasets-facets-in-open-was-google-refine

Advanced refine

Calling REST services and parsing JSON with OpenRefine: https://github.com/DruidSmith/OpenRefine-Recipes/blob/master/URLs_and_JSON.md

Extending Data with OpenRefine: http://www.xavigimenez.net/blog/2013/08/extending-data-with-openrefine-ex-google-refine/

Improving Access to the Lawrence Collection (extending collection data with OpenRefine and Freebase): http://www.nli.ie/blog/index.php/2012/04/05/improving-access-to-the-lawrence-collection/

Reconcilable Data Sources for Librarians: https://github.com/OpenRefine/OpenRefine/wiki/Reconcilable-Data-Sources#librarians

Cheat sheets and specific solutions

Analyzing Usage Logs with OpenRefine: http://acrl.ala.org/techconnect/?p=4253

Google Refine Cheat Sheets (includes Regular Expressions): http://arcadiafalcone.net/GoogleRefineCheatSheets.pdf

Clean Up: Dates and OpenRefine: https://icantiemyownshoes.wordpress.com/2014/04/24/clean-up-dates-and-openrefine/

A Simple OpenRefine Example: Tidying Cut n' Paste data from a Web Page: http://blog.ouseful.info/2013/05/01/a-simple-openrefine-example-tidying-cutnpaste-data-from-a-web-page/

Data Shaping in Google Refine-Generating New Rows from Multiple Values in a Single Column: http://blog.ouseful.info/2012/07/30/data-shaping-in-google-refine-generating-new-rows-from-multiple-values-in-a-single-column/

Comparing Columns in Open Refine: http://blog.ouseful.info/2011/08/06/comparing-columns-in-google-refine/

Merging Datasets with Common Columns in OpenRefine: http://blog.ouseful.info/2011/05/06/merging-datesets-with-common-columns-in-google-refine/

References

1. About GOKb (2016). Retrieved from http://gokb.org/about-gokb/
2. Becknell, E., & Weeks, S. (2013). *Using OpenRefine to update, clean up, and link your metadata to the wider world* [PowerPoint slides]. Retrieved from http://downloads.alcts.ala.org/ce/09182013_Using_Open_Refine.pdf
3. Bedoya, J. (2014, May 7). Analyzing usage logs with OpenRefine. *ALA TechConnect* [Web log post]. Retrieved from http://acrl.ala.org/techconnect/?p=4253
4. Heller, M. (2013, May 1). A librarian's guide to OpenRefine. *ALA TechConnect* [Web log post]. Retrieved from http://acrl.ala.org/techconnect/post/a-librarians-guide-to-openrefine
5. Magdmartin, M. (2014, August 29). 2014 survey results. *OpenRefine* [Web log post]. Retrieved from http://openrefine.org/2014/08/29/2014-survey-results.html
6. Samples, J. (2014, April). *Cleaning the metadata mess: Using OpenRefine to transform and share your library's data*. PowerPoint presentation at the Electronic Resources and Libraries Conference, Austin, TX.