Open Reconcile: A Practical Open-sourced Ontology-driven Webservice

Rebecca Lawler and Hui Yang
Computer Science Department
San Francisco State University
San Francisco, California
r.lawler@gmail.com and huiyang@sfsu.edu

Kathryn Woods and Josh Kaminker
Bioinformatics and Computational Biology
Genentech
South San Francisco, California
woods.kathryn@gene.com and kaminker.josh@gene.com

Abstract—Curators in specialized fields such as biotechnology, often have to either rely on in-house tools or do tedious tasks manually. To address these issues, we have implemented Open Reconcile, an open-source and general reconciliation tool that ensures the compliance of a dataset to a specific controlled vocabulary. Open Reconcile is compatible with the Google Refine Reconciliation API, where Google Refine is a tool for data analysis. Open Reconcile is highly customizable and supports data from different database applications. It adopts multiple strategies to find the optimal match to reconcile input terms with those in a controlled vocabulary. It also allows users to configure a synonym table to facilitate auto-corrections that can only be performed with the support of domain expertise.

Keywords—controlled vocabulary, enterprise ontology, data reconciliation, data cleaning, web services

I. INTRODUCTION

Biotechnology companies nowadays frequently follow a data-driven approach to formulate new hypotheses and suggest new experiments [1]. To this end, it is of paramount importance to ensure the cleanliness, integrity and consistency of company-owned data. This will enable effective and smooth collaborative exploration among scientists from different departments in a multi-disciplinary enterprise environment. Due to lack of mature ontologies (e.g., SNOMED [18]) or controlled vocabularies (e.g., UMLS Metathesaurus [19]) in the past, well-established biotechnology companies often create their own ontologies or controlled vocabularies to impose a “universal terminology” over the entire company, thereby ensuring the naming consistency of entities such as cell lines, tissue samples, diseases, and drugs. Consequently, professional curators are often actively involved at biotech companies to maintain the cleanliness and integrity of such controlled vocabularies.

These curators primarily focus on the following two types of curation tasks:

(1) Data cleaning and reconciliation: For a new set of incoming tissue samples, the curators in charge need to scrutinize the entries to deal with spelling variations (e.g., “Oestrogen” vs. “Estrogen”) and abbreviations (e.g., H.Sapiens versus Homo Sapiens). A more important goal here is to reconcile the disease and tissue names mentioned in this incoming dataset consistent with those in the company’s controlled vocabularies.

(2) Maintaining the controlled vocabularies: The goal of this task is twofold. First, the curators in charge need to locate and fix the entries that have been entered into the vocabularies due to haste or a pressing deadline. Second, these curators examine the semantically related entries and determine whether to substitute them with another term. For instance, the terms “cecum” and “rectum” can be replaced by “colon”. The term “abdominal cavity” should be changed to “abdominal wall” if organism location is of major interest.

Without an effective software tool, these tasks routinely take curators days or maybe weeks. To improve the efficiency, curators often resort to writing mono-functional programs. They however face a major dilemma: should they spend their time in coding or curating? Although most curators know some basic scripting programming languages such as Perl, coding is not their specialty. The extreme scenario can be that a curator ends up writing a new script for each task, which in some cases virtually negates any advantage brought forth by coding over doing it manually. More importantly, a curator’s time is best spent on areas where he can do better than a computer program, for example, making the decision of replacing “abdominal cavity” with “abdominal wall”. It is therefore desirable and critical to bring in computer scientists as part of the solution.

Inevitably, computer scientists face a different kind of dilemma: should they use a publicly available software tool or develop an in-house tool? The answer to this question depends on the specific requirements of the above two curation tasks. The primary goal behind these requirements is to significantly reduce the high cost of curation. Specifically, the curators have the following requirements for the software tool:

- This software needs to be able to automate or semi-automate the common routine subtasks encompassed by the two primary curation tasks. An example of such a subtask is to identify different orthographic
features such as abbreviations, capitalization, and use of numerical values.

- This software tool needs to be equipped with a user-friendly and familiar graphic user interface (GUI). Specifically, curators prefer an interface similar to the spreadsheet-like layout of Microsoft Excel, a de facto standard for scientists in the biotech industry.

- Due to the special characteristics of curation, this software is not expected to completely replace the human curators. Instead, it should provide entry points for a curator to interact with the automatically rendered data items and make suggestions or corrections based on his domain expertise.

- This software needs to include an interface to access the controlled vocabularies, which have been typically managed by the Oracle database system.

   From the perspective of a computer scientist, it is also a desirable, but not necessarily a required feature if this software tool is also applicable to other application domains, for instance, historical artifact curation at a museum.

   Several existing software tools can help make the data cleaning and reconciliation task less burdensome. Such tools include DBWiki [2], MetaWeb [3], and Google Refine [5]. DBWiki is designed to target the data-cleaning task. MetaWeb provides a gaming environment aiming to make the classification task less tedious. Finally, Google Refine deals with data reconciliation. However, none of these tools meet all the above requirements.

   Specifically, DBWiki works by exporting information from the desired database into a Wiki-style output. It allows a more intuitive way to look through not only the data, but also a history of the changes made to such data. DBWiki is however not sufficient to address the cleaning task faced by the curators as it doesn’t provide functionality beyond representing the data in a database.

   MetaWeb on the other hand turns the categorization task into a game for curators to play [4]. While this is novel and can increase curators’ productivity, it would not help accomplish the related curation task other than possibly making it less tedious. In addition, MetaWeb’s software is not open source and specific to their Freebase database.

   Google Refine is an open source web-based application that is “a powerful tool for working with messy data” [5]. It is a desktop application that consists of a web-based user interface with several tools for bulk data management. The tools include: text faceting with histogram analysis, regular expression support, and clustering based on nearest neighbors. These tools would be useful for a curator who deals with large amounts of data and needs to get a good understanding of the composition of such data. Google Refine also features the ability for a user to reconcile an individual column either against MetaWeb’s Freebase database or user-defined databases, though the latter requires the developer to implement a new web service.

   On the surface, Google Refine seems a sufficient tool for the two curation tasks identified earlier. It has built in regular expression scripting support as well as a GUI that would be familiar to most Microsoft Excel users. It has additional features that are very powerful. For example curators can use the text faceting feature to visualize the data. They can use clustering to detect potential mistakes such as typos and capitalization. The reconciliation feature also seems like a good fit for some tasks.

   However, when one delves into the different functional features of Google Refine, its limitations become apparent. First, its reconciliation feature is useful at reconciling general standardized information such as species names, but is of little use at reconciling specialized, non-standardized information such as canonical cell line names. These names are often sensitive to the use of lower/upper case and punctuations (e.g., hyphen). Second, although Google Refine contains a module to find semantically similar terms, a function required by the second curation task, it is fully automated and does not accept inputs from curators. (This is important for curators in biotech.) Finally, Google Refine by default reconciles terms with MetaWeb’s Freebase. Freebase is a product of MetaWeb, which is a massive database of knowledge (similar to Wikipedia only with more meta analysis of data). Although Freebase has a publicly available API for software developers, it is primarily set up as data storage facility for publically available data. It is therefore inappropriate for biotech companies to use on proprietary data if privacy is important. In other words, the reconciliation service needs to reconcile against the company’s own proprietary databases, which are Oracle databases in our case. (Note that since Google Refine has optional extension packages by a third party that offers additional options that allow reconciliation against RDF (Resource Description Framework) [6][7], one might argue for the solution to convert the Oracle database to RDF. This is too time-consuming an option for our curators because they have to convert the database whenever a change is made to the database, which may be quite frequent. They will also need to load the RDF data back to the Oracle database for each dataset.)

   Fortunately, Google Refine provides flexible APIs that allow developers to address the above limitations. Specifically, we have implemented an open source, web-based reconciliation tool, Open Reconcile. Open Reconcile allows curators not only to take advantage of Google Refine’s existing features that are applicable to our curation tasks, but also customizes Google Refine to meet the curators’ specific needs as discussed earlier.

   As a curation tool, Open Reconcile enables Google Refine to meet all the specific requirements discussed earlier. Below is a summary of the main functions of Open Reconcile:

   - It implements a reconciliation service that supports Oracle [8] and Postgres [9] database in addition to Google Refine’s default settings.
   - It includes a front-end tool that allows end-users to conveniently specify the location information of a
controlled vocabulary. This function is especially useful for biotech companies as they frequently develop or introduce new vocabularies. It also allows users to set some basic matching rules about sensitivity (e.g., whether matches are case- and/or punctuation-sensitive).

- It allows curators to establish semantically or hierarchically related terms through a user-friendly GUI. This feature allows the integration of human knowledge into the curating process. For instance, the curator can use this feature to indicate that “cecum” and “rectum” should be generalized to “colon”.
- It also allows multiple controlled vocabularies from different sources to be contained within the same reconciliation service. This will give the curator the flexibility to choose from different controlled vocabularies for reconciliation.

Our professional curators have quantitatively and qualitatively evaluated Open Reconcile in the production environment. The evaluations demonstrate that Open Reconcile is effective at significantly cutting curators’ workload. In several cases, the employment of Open Reconcile reduced the curation time from several days to less than one hour. We are currently in the process of deploying Open Reconcile.

II. SOLUTION

A. Google Refine-based Requirements

Because Open Reconcile is designed to function seamlessly with Google Refine, it is important to understand the reconcile command employed in Google Refine. (See the previous section for a brief introduction to Google Refine as a software product.) Reconcile is a column-based operation. Selecting this operation opens a dialog box that prompts the user to select the desired service and the desired type from this service.

After a type is chosen, the user instructs Google Refine to reconcile. The reconcile program does this by sending requests in batches to the reconcile service. Each batch consists of an array of up to ten queries. It is posted to the selected server via a JSONP request, where JSONP is a variation of JSON (JavaScript Object Notation) intended for cross-domain communication. This request enumerates not only the terms queried for, but also the type specified and other additional information to be used in formulating the response. Here are the fields specified in the API for a query:

- “query” - the term to be searched for.
- “limit” - the maximum number of results to return
- “type” – the type that the term should be reconciled against.
- “type_strict” – an optional field which is not used in this solution. It controls how strictly the “type” should be obeyed
- “properties” – an array of user selected data to be sent with the query. They can be useful depending on the type of data and how interconnected these data are, but are optional. Since one of the design goals of Open Reconcile is to be content-agnostic, support for this field is not included.

To respond to each query, a JSONP response is constructed through the HttpResponse mechanism. It adds a callback field to allow the response to comply with the Site of Origin Policy. In addition to the callback, the returned JSON must also consist of the following information for the returned results to each query (there may be between zero to many responses for each query):

- “id” - a string for identification. It should be unique for each response
- “name” - the name of the term to which the query has been matched.
- “type” - an array of types that a query has matched with. In the current version of Open Reconcile, it will always only be one term, which is the term requested by the query.
- “score” - a double value indicating how well the returned result matches the term under query.
- “match” - a Boolean value that denotes if it is an exact match or not. If it is not, Google Refine will allow the user to make a judgment before the cell is marked as reconciled

In a completely clean data set, all reconciled cells should receive results where the “match” field is returned as “true”. In a completely dirty data set, the “matched” field should be false. The faceting functionality in Google Refine allows users to filter results depending on whether an exact match is returned. The cells without an exact match will have the original value presented along with the suggested partial matches. A user can then select a partial match as correct and use it to replace the original value.

In the case of Open Reconcile, because it is designed for biotech data, if there is any doubt, a human judgment will be requested. While false-positives are not acceptable, too many false negatives would defeat the purpose of Open Reconcile as well. We will discuss how we avoid false positives and false negatives in the next section.

B. The Overall Architecture of Open Reconcile

Open Reconcile is a web-based service that consists of a client and server. The client, or the front end, is used to collect configuration information from user input. This data is stored in a configuration file. The configuration file is read by the server, or the reconciliation servlet, and allows a user to customize how the reconciliation service behaves without having to modify the source code, or even restart the service.

While designing Open Reconcile, there are many goals aimed at maximizing its usefulness and flexibility. One of the goals is to create a product that people could easily use.
Although it is born of corporate needs, it is intended to also potentially be of use to other users, including hobbyists and academics.

C. The Back-end Solution
The goals of the back-end (or server) portion of Open Reconcile are the following:

- It should be able to run on a local machine with the proper software installed.
- It should not require any commercial software.
- It should be easily distributed in a package format.
- It should be functionally content-agnostic.
- It should be able to be configured by the user, both in terms of data source and in terms of matching rules.
- The configuration by the user should be stored on the server in a file and should not expire. The configuration file should be able to be written to by the front-end and read from by the reconciliation service.
- The design and code should be as efficient as possible with as few database operations as possible, but should not compromise the advantage of accessing the database for fresh information.

The configuration file is stored on the server. It is an SQLite flat file, which is chosen for its lightweight database properties. It will also allow access from multiple users without resulting in data inconsistencies at a relatively low overhead.

The following information is stored in the configuration file:

- A unique type name.
- Database connection information:
  - data source type (Oracle or Postgres)
  - server name
  - port number
  - database name or service name
- Data source details:
  - table name
  - column name
- Boolean values to indicate if punctuation and capitalization should be considered in comparisons.

The information above is necessary for the reconciliation service to function properly.

To handle cases that request human input, a synonym table was added to Open Reconcile. Strictly speaking, this table does not contain only synonymous terms. It contains terms that are either semantically similar or hierarchically related. As an example, the curators may deem it necessary to replace “human” by “homo sapiens” during reconciliation. To do this, he can simply add these two terms to the synonym table.

The synonym table is stored in the same configuration file, but in a different table. It contains the following information:

- The string value to be replaced (e.g., “human”).
- The corresponding value to replace the above value (e.g., “homo sapiens”).
- The type this rule is to be applied for (e.g., “species name”).

Open Reconcile is implemented entirely in Java. It is distributed as a WAR package, and runs on common server software such as Tomcat.

The configuration file is at the heart of the reconciliation service. Data from the configuration file is used to query information from the database, to generate and score the matching results, as well as to generate metadata. Metadata is a list of types available for reconciliation and is used by Google Refine to formulate the queries sent to the service.

Figure 1 depicts the overall architecture of the back-end of Open Reconcile.

![Figure 1 The Back-End Architecture of Open Reconcile.](image)

Figure 1 - Reconciliation Sequence of Events
The queries sent to the service are put through the matching class, which matches a given term with those in the controlled vocabulary. This matching class is designed to be content agnostic. In other words, it does not depend on any special domain knowledge. Its input data is the configuration file, including synonym data. It also handles a variety of error-cases such as typos.

The algorithm employed by the matching class uses a variety of methods following a particular sequence of events.

The first event compares the query string with the values available in the synonym table. If a match is found, it is replaced. The following matching procedure, as shown in Figure 2, is then applied to reconcile the replaced term.

As depicted in Figure 2, the matching procedure first searches for a perfect match. It will stop the matching procedure if a perfect match is found. A perfect match depends on the configuration. For example, if punctuation and capitalization are ignored, a non-exact match may be considered as a perfect match if the only differences are capitalization and punctuation characters. In cases where there is no perfect match, the following steps will be applied. A vocabulary term will be considered a partial match if:

- It contains part of the query term or is part of the query term.
- The edit distance is below a certain threshold.
- If the query term contains multiple words (separated by spaces) and any of the following criteria is true for one of words.

Given a term for reconciliation, the above procedure will return either a perfect match or a set of partial matches. These partial matches are sorted by their matching score and pruned according to the number specified in the query request. The scoring function in the matching class indicates how well a term matches with terms in a controlled vocabulary. A JSONP response containing the results is then created and sent back to Google Refine.

D. The Front-end Solution

Permitting user-configurable data is essential to Open Reconcile. A straightforward front-end is implemented to facilitate easy configuration. The front-end communicates with the back-end through the configuration file as shown in Figures 1 and 3. The front-end accepts user input and saves it in the configuration file.

The front-end is designed to meet the following criteria:

- Using simple forms to collect user input.
- Allowing the user to view and edit the data he enters.
- Preventing the user from entering unusable data by suggesting corrections when feasible.

Figure 3 presents a schematic description of the front-end.

![Front-End Architecture](image)

**Figure 3 - Front-End Architecture**

Google Web Toolkit [14] was used in the development of the front end for the following reasons:

- It comes with standard support and themes for common conventional website widgets.
- Libraries exist to add further polish to the UI experience, for instance, with animations when waiting for a response.
- It is Java-based, lending itself well to modular code development.
- Objects can be passed between the client and the server, allowing for strong data encapsulation.
- It handles calls to the server in an asynchronous manner.

Two separate configuration pages are developed. One page was created for adding a new type, the other for modifying existing types.

The user may add a new type by using either the wizard or the manual entry method (Figure 4). The wizard method guides the user through table name and column name options. The manual method spawns text fields for all of the desired data. After the configuration information is entered and the “Add” button is selected, the service checks to ensure the table and column are accessible and then adds them to the configuration file.

The front-end also provides an option that allows the user to edit the current types typed earlier. Figure 5 shows the user interface. The user can choose to delete a type, to preview a subset of the terms of that type, and to view the database connection information for that type. The information in the synonym table can be added or deleted as well through this interface.

Because of the type-dependent nature of the synonym table, the configuration option for it is located on the Configure Current Types page, and can only be accessed for one type at a time.
III. RESULTS

The goal of Open Reconcile is to support different types of curation tasks as described in Section I. Two curation tasks have been evaluated using a comparative approach. In other words, the same curation tasks were performed with and without the support of Open Reconcile.

The first task is cleaning dirty data to test the matching algorithm. The data set used is a set of canonical cell line names, about four thousand terms long. While it does contain similar entries, it contains no identical terms. Because of the nature of canonical cell line names, case and punctuation sensitivity were both selected during configuration. It was reconciled using Open Reconciliation in under 15 minutes.

To compare the results, the hand-curated version was also reconciled. Because of the nature of the data, a match-rate of 100% was not expected for either data set. The results are promising. There were no false positive matches. There was an 89% match rate, very close to the same rate as the hand-curated version (90%). The matched terms from both lists were virtually identical (the non-hand curated list had terms that were not on the curated list).

Of the 437 samples that were not matched, 201 contained the correct matching term. The most common reason for failing to reconcile the correct term as a match is the presence of dashes in place of spaces (e.g. “A 704” versus “A-704”). Of the results that did not include the match, 180 samples contained the name of a related cell line. Only 54 did not contain any related suggestions. In this case, these samples all have very short names, e.g., “ST”. Lastly, 4 samples did not have any results suggested by Open Reconcile.

The most significant shortcoming of the matching algorithm in this test is unrelated suggestions. The algorithm would be improved if it were better at identifying terms that are not in the controlled vocabulary set. However, returning unrelated results is preferred to returning too few results.

The second test was a “roll-up” task. This task allows scientists to look at samples from a “birds-eye view.” It is a consolidation of similar types (for example, classifying “adenocarcinoma” and “carcinoma” as both being “Cancer”). The dataset used is a set of tissue type classifications for over one thousand samples. A domain expert supplied the rules for the roll-up. There were 78 rules.

A table was created populated by the desired end classifications, and a type for the data was configured. The mapped values (“Rectum” to “Colon”) were entered into the synonym table for that type. The column was then reconciled against that type. Any non-matching results would be interpreted a mistake, as all values should be converted to one compatible with the data type.

The results of the synonym table were very strong as well. Without error, 100.0% of the terms reconciled with 100% matches (1202 out of 1202 non-blank terms successfully reconciled). A curator completed this task using Perl scripting, over the course of about two days. While data entry by hand for the synonym table is a tedious task it took under an hour, and the reconciliation command executed in a matter of minutes.

The only disadvantage of this method over the traditional scripting is that with scripting all of the blank values could be easily rolled-up to “Unclassified”, whereas they are not sent for reconciliation in Google Refine. Another method is needed to accomplish this.

In both tests, the quality of the results was comparable to those of curators using traditional but time-consuming methods. The significant advantage of using Open Reconcile thus lies in its time saving capability as a flexible web-based curation tool.

After having used Open Reconcile, a professional in-house curator stated that, in his experience, Open Reconcile filled a gap where no standard tool is available. This curator is able to use Open Reconcile after a short training session.
with the first author of this article. He especially enjoys the following functionalities of Open Reconcile: (1) the simple and intuitive graphic user interface, (2) the ability to create his own synonym table to influence the reconciliation process, and (3) the huge amount of time he is able to save by using Open Reconcile.

IV. RELATED WORKS

Google Refine is an open source framework to handle messy data. A number of its APIs have been published via Google Code. It allows for the creation of extensions through Butterfly’s extension framework.

Presently, the following extensions have been realized for Google Refine and are available in the public domain:

- RDF extension [6], which is a powerful extension that makes use of the relational data framework. It not only accepts the import of RDF dumps and SPARQL endpoints as sources to reconcile against, but also allows Google Refine to export the cleaned data in RDF format.
- VIVO extension [9], which allows for reconciliation against VIVO’s data format using an HttpServlet. It relies on RDF data, and can be considered an extension to the above RDF extension.
- Statistics extension by Chicago Tribune [11], which is in response to the interest in Google Refine by data journalists. It allows for import and export from Postgres data sources, but regrettably has not been maintained since 2010.

A couple of third party reconciliation services have also been developed:

- OpenCorporates, which is a Python-based standard reconciliation service [12]. It is among the first publically available non-Freebase reconciliation services. It is however not open-source.
- Helmut, which is a Python–based, open-source reconciliation service [13]. It is based on OpenCorporates’ service. It runs locally using a flask server and stores the data source in a webstore.

All of these tools are useful and do address some of the concerns that Open Reconcile does, but they are not good long-term solutions. For instance, many of these extensions support locally run services. They are not appropriate for many curation tasks where multiple curators often need to simultaneously work on the same controlled vocabulary to ensure internal consistency.

Additionally, none of these solutions offers any type of customization for particular types of terms. Instead, they apply rules universally over all the data types on the server.

Finally, as discussed earlier, converting data into RDF dumps or other formats are too time consuming for database that are regularly updated.

In contrast, Open Reconcile allows more flexibility. For example, it allows users to specify rules for different types of terms. It also supports multiple data source types such as Oracle and Postgres databases, and could be easily extended for additional support. Note that it is not only for immediate business needs that we choose Oracle as the database to support, but also because of the strong presence of Oracle in enterprise database solutions [16].

V. DISCUSSION AND CONCLUSION

The quantitative evaluation reported in the Section III demonstrates that Open Reconcile is a promising curation tool. Professional in-house curators welcome such a flexible and extensible tool as it bridges the gap between standard tools and urgent needs for efficient curation. Open Reconcile however is by no means perfect. There is still a great deal of room for improvement. We next discuss a few such cases.

The synonym table works well, but it can be time consuming to enter a large list. In some cases it is beneficial to modify the configuration file directly with a third party tool, rather than through the UI. This type of situation should be rare, but it could be addressed through an added feature.

Open Reconcile makes a number of assumptions that limit its functionality. It only supports populating a type by using all entries from one column. It may be useful to allow users to select multiple columns. Also, it only searches for the public schema for table names in Postgres. It is geared towards data sets that are comprised of one-word terms much more often than multiple word terms. Performance issues are foreseeable if it is used on a big list of long terms.

Open Reconcile is also limited by the database technology it supports. It would benefit for additional support for database systems with growing popularity, such as Microsoft SQL Server [17].

During a correspondence with the creator of helmut, Friedrich Lindenberg, he expressed an interest in making Open Reconcile capable of populating a local data store using data from a CSV file. Other Google Refine-focused discussions with the Google Group for the project, demonstrated a significant amount of enthusiasm for the project, especially altering it to be customized for smaller tasks. While these applications do not have much appeal for an enterprise-oriented solution, it shows would be useful to users who need more flexibility.

Additionally, Open Reconcile does not take advantage of the rich and complex functionality of Google Refine’s Reconciliation API. It currently does not support the handling of complex semantic relationships and the inclusion of additional data. Such functions will be useful.

In conclusion, Google Refine is a growing technology with many powerful features for curators. Open Reconcile extends the functionality of Google Refine, bridging the gap between open source functionality and business need. Our evaluation results show that Open Reconcile is valuable at significantly increase the curation productivity.

VI. REFERENCES
