IJDC | *Peer-Reviewed Paper*

Amplifying Data Curation Efforts to Improve the Quality of Life Science Data

Mariam Alqasab
School of Computer Science
University of Manchester

Suzanne M. Embury
School of Computer Science
University of Manchester

Sandra de F. Mendes Sampaio
School of Computer Science
University of Manchester

Abstract

In the era of data science, data sets are shared widely and used for many purposes unforeseen by the original creators of the data. In this context, defects in data sets can have far reaching consequences, spreading from data set to data set, and affecting the consumers of data in ways that are hard to predict or quantify. Some form of waste is often the result. For example, scientists using defective data to propose hypotheses for experimentation may waste their limited wet lab resources chasing the wrong experimental targets. Scarce drug trial resources may be used to test drugs that actually have little chance of giving a cure.

Because of the potential real world costs, database owners care about providing high quality data. Automated curation tools can be used to an extent to discover and correct some forms of defect. However, in some areas human curation, performed by highly-trained domain experts, is needed to ensure that the data represents our current interpretation of reality accurately. Human curators are expensive, and there is far more curation work to be done than there are curators available to perform it. Tools and techniques are needed to enable the full value to be obtained from the curation effort currently available.

In this paper, we explore one possible approach to maximising the value obtained from human curators, by automatically extracting information about data defects and corrections from the work that the curators do. This information is packaged in a source independent form, to allow it to be used by the owners of other databases (for which human curation effort is not available or is insufficient). This amplifies the efforts of the human curators, allowing their work to be applied to other sources, without requiring any additional effort or change in their processes or tool sets. We show that this approach can discover significant numbers of defects, which can also be found in other sources.

Received 17 October 2016 | *Revision received* 7 June 2017 | *Accepted* 20 June 2017

Correspondence should be addressed to Mariam Alqasab, School of Computer Science, Oxford Road, Manchester, M13 9PL. Email: mariam.alqasab@postgrad.manchester.ac.uk

An earlier version of this paper was presented at the 12th International Digital Curation Conference.

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Copyright rests with the authors. This work is released under a Creative Commons Attribution 4.0 International Licence. For details please see http://creativecommons.org/licenses/by/4.0/

Introduction

Many database owners make their data available on the Web for others to use. While in most cases no assurance is provided regarding the quality of Web data, data is generally considered trustworthy if its quality level is high. A key part of this task is making sure that data stored in the database is an accurate reflection of reality, but other aspects of data quality should also be taken into account, such as completeness, consistency, believability and currency.

Data in databases are not fixed: database providers will make changes as and when they are required, to mirror the part of the real world modelled by the database. To keep data quality high, database providers use automatic and manual curation techniques to find and remove defects in data. Some of these defects can be detected using automatic tools, though these tools tend to be limited to the detection of systematic errors of a simple and generally applicable form. An example would be a tool for completeness checking that determines whether all required fields contain data and warns the user about those that are empty. However, these automatic tools tend to be limited to systematic errors, leaving the detection of non-systematic errors unsolved. Typically, non-systematic errors require knowledge and expertise in the domain to be detected and fixed. For example, in biomedical area, errors in the recorded function of a protein need a human expert and detailed analysis to discover by examination of new publications.

In many domains, this expertise is not widely available and is not free. Domain experts have to spend a lot of time searching for defects in data and fixing them. Some experts may volunteer their time to curate data in their domain of interest; in other areas, owners of core community resources have to hire data curators to maintain the accuracy, concurrency, completeness and consistency of their data. For example, the UniProt database[1] has a number of data curators who are paid full-time to manually curate UniProt data. However, there are many smaller data sources that are managed on a voluntary basis and have no funding to employ data curation. These databases can become out-of-date over time.

We would like to find an automated way to package data curation work in a form that can be cheaply and easily applied to these other databases and data consumers without requiring any extra work from the data curators. In this paper we present IQBot, an approach to solving this problem. Our aim is to find a way to detect data updates made by curators, determine the reason behind the change and share those that may have value more widely. We also present experimental results confirming that IQBot can find defects that are also present in other databases.

Related Work

Since database consumers care about using high quality data to produce correct results, database providers must also care about data quality and must try to keep it as high as their (often scarce) resources allow. This has lead to research on data curation ranging from proposals for semi-automatic tools requiring interaction from human experts, to

[1] UniProt: http://www.uniprot.org

completely automated curation approaches.

Abrams et al. provided a tool for database providers to share their data with others by storing them in a Merritt repository[2]. Each time a database provider curates their data, they update the version stored in the repository. By updating the repository version, database users are able to get the most up-to-date version of the data. A graphical user interface helps data consumers to find and download datasets that match their needs (Abrams et al., 2014).

Ravagli et al. proposed OntoBrowser, a collaborative tool for the curation of ontologies. The tool allows curators to work with a single copy of data, stored in a central database to avoid redundancy (Ravagli, Pognan and Marc, 2016). The tool can also pre-map unmapped ontology terms automatically by using fuzzy matching.

Some curation tools are not fully automatic, but require humans to participate in the curation process. Bunt et al. proposed a semi-automatic curation tool that sends e-mails to authors who have a new publication in the area, and asks them to fill in a specially designed form. In return, the provided information helps curators to speed up the curation process, as they will have all the basic information needed to easily identify the data to be curated (Bunt et al., 2012). However, authors responses to these e-mails are voluntary. There is no guarantee that all authors emailed by the tool will response and participate in the process.

Other research proposed different approaches for automatic data curation tools. Kumari et al. evaluated tools supporting on-demand curation by using a case study to see how the presentation of information about uncertainty of data can affect the performance of the curation. Their study showed that the way in which the uncertainty was presented affected the way users interpreted the results. The authors suggested user interface guidelines for the presentation of uncertainty to curators to aid in efficient and accurate curation decisions (Kumari, Achmiz and Kennedy, 2016).

Some researchers have tried to find ways to minimize the time needed for data curation by reusing and sharing curation efforts with others. Orchard et al. proposed the International Molecular Exchange (IMEx), which allows the participant protein databases to pool their literature curation efforts (Orchard et al., 2012). Redundant curation work is prevented by the rather coarse-grained mechanism of assigning certain academic journals to each partner organisation. Each partner extracts protein interaction records only from the journals to which they are assigned. The curators also need to follow a set of curation rules to make the work applicable to other databases. Another research project, called MIntAct, also focused on sharing curation efforts between databases by providing a common curation platform (Orchard et al., 2013). Although both IMEx and MIntAct share curation efforts with other databases, the sharing of curation efforts is limited between the databases run by the partner organisations.

As this brief survey shows, the research literature contains some proposals for curation tools aimed at improving data quality, including some efforts with a similar aim to the research described in this paper: the aim of making the benefits of limited curation efforts stretch beyond the direct users of the database under curation. However, at present, approaches to sharing curation effort are limited to groups of participants who agree in advance to collaborate and use the same curation tools. In our own work we aim to create bridges between the curated data sources and users of that same data in other, unconnected

[2] UC3 Merritt Repository: http://www.cdlib.org/services/uc3/merritt/

databases. In addition, our approach does not require data curators to change their tool set or make any additional effort to collaborate with others.

Extracting Defects from Changes

To achieve our aim of increasing the value of data curation work, we propose a component called an IQBot. IQBots monitor curated data sources, looking for the changes the curators make from one version of the data to the next. These changes were made to address some problem a curator perceived in the data (to add missing data or correct errors, or change the representation of data to one that is more useful to current data consumers, for example). These changes are a useful source of information on the quality of the data. The IQBot examines each change, attempts to infer the reason for the change, and exports any changes deemed to be of general interest for sharing with other data owners. In this section, we explain how the general IQBot component works. Afterwards, we will explain how this general component is instantiated to allow it to gather defect and correction information from a real database.

The main tasks of the IQBot are: to detect changes made to data in the monitored database, to infer the reason for the changes (i.e. to infer when a change is a result of a generally useful defect), to decide from that reason whether the change is of wider interest, and to publish it as a defect-correction pair. An IQBot does not only focus on the most recent changes made to data. If the curated data source maintains a history of versions, the IQBot can extract the full history of curator-led changes over the life time of the source.

Extracting changes using IQBot

Algorithm 1 shows the main processing behaviour of IQBot as pseudocode. IQBot starts by comparing the version of the data specified by the owner (typically the latest version) for changes. If this is the first time this source has been monitored, then IQBot will work back through all the versions, looking for and publishing changes. However, in normal use only the two most recent versions will be compared.

As the IQBot's job is to extract changes in data by comparing the contents of consecutive versions, the curated database needs to have at least two accessible versions of data. If the data source to be monitored does not explicitly provide access to version information, the owner of the IQBot can arrange to take periodic snapshots of the source, and to compare them for changes (Embury, Jin, Sampaio and Eleftheriou, 2014). It should be further noted that it is not part of the IQBot's job to check whether the assigned curated database is versioned or not. This type of information is provided by the publisher of the curated database and it is outside the scope of this paper.

When comparing two versions, the IQBot first attempts to locate the IDs of all records that have been modified in the version being monitored. If the monitored source offers versioning facilities, it may be possible to query for this set of record IDs directly. If it does not, IQBot will extract the full set of IDs in both versions, and will check each one for changes (this is slow, but IQBot is not intended to work in time-critical applications). To extract these IDs (and the details of the records they refer to), IQBot needs to be configured to access the data source being monitored, through whatever API is provided

Algorithm 1 The general algorithm of the IQBot

```
Require: v : the version of the data set we are monitoring for changes
 1: vPrev = predecessor version to v
 2: while vPrev exists do
 3:     if v has not yet been monitored then
 4:         changedRecs = get IDs of records changed between v and vPrev
 5:         for each recID in changedRecs do
 6:             recV1 = read the record with id recID in version v of the data
 7:             recV0 = read the record with id recID in version vPrev of the data
 8:             if recV1 != recV0 then
 9:                 type = findChangeType(recV1, recV0)
10:                 reason = findTheReasonForTheChange(type, recV1, recV0)
11:                 publish(recID, recV1, recV0, reason)
12:             end if
13:         end for
14:         mark version v as having been monitored
15:     end if
16:     v = vPrev
17:     vPrev = predecessor version to v
18: end while
```

for programmatic access to data. If necessary, the data may need to be downloaded and hosted locally, at the site where the IQBot is running. A plug-in must be provided for IQBot, allowing it to execute the data access queries it needs to do its work against the monitored source, and to insulate IQBot from differences in data model/representation (e.g. relational tables vs text files).

IQBot then compares the versions of individual records, looking for details of what has changed, to identify the type of the change. Records may be added or deleted, attribute values may be changed, added to or deleted from. Records may be merged or split into two. Different defects will require different types of change to address them, and therefore the type of change gives a clue as to the defect that the change corrects.

A further issue to consider is the type of change that should be monitored. Curators may make a great many changes of different sorts to different parts of the data. Not all of these may be of wider interest. As with other aspects of the IQBot, the owner can decide which changes are to be extracted by defining a plug-in component, indicating which parts of the schema are to be monitored for which types of change.

Finding the Reason for the Change

When the IQBot detects a change in data, then the reason behind the change needs to be inferred. However, the changes first need to be classified. To do this, we divided the changes into a number of types: simple, partial and complete changes. We identified these types by observing a collection of 1499 changes detected by the IQBot in a sample curated data source.

Simple changes are those which change the form of a represented value, but not its content or meaning. Examples are corrections to spelling mistakes, changes of punctuation and changes of letter case. Such changes are normally made to preserve consistency within the curated database, or because of changing conventions used by curators. We hypothesise that these changes are not of great interest beyond the curated source itself.

Partial and complete changes both involve a change of content or meaning. Partial changes are applied in only some of the places in which the change could be applied, while complete changes cover the entire data set. They occur when curators need to make corrections to recorded measurements, and updates to interpretation data, to reflect the changing understanding of the scientific field covered by the data source. The reasons behind them are typically highly domain specific, and typically cannot be generalised to apply to different databases in other areas. To allow IQBot to infer the reasons for these changes, a plug-in component must be provided.

The UniProt Protein Names IQBot

As we have seen, to create a functioning IQBot, it is necessary to configure the general component with plug-in components providing the required domain and system specific logic. To test our ideas, we created an IQBot to monitor changes made to UniProt: the Universal Protein resource. UniProt is a core reference for biomedical scientists. It contains around 66 million protein entries, some manually curated and some automatically curated through software tools. Due to the importance of the data and the scale of the curation task, UniProt employs a team of expert data curators. Their job is to read and analyse all the publication in the area, and apply their knowledge in order to detect and fix defects in data to provide high data quality for the consumers.

We selected UniProt as the monitored curated database for the following reasons:

- UniProt data is extensively manually curated by domain experts.
- The data is continuously curated, with new releases being frequent (every four weeks). This allows us to gather experimental data frequently.
- A long history of previous versions can be accessed, allowing us to experiment with the use of IQBot on a long data lifetime.
- UniProt has a programmatic interface, through which data can be accessed.

In the previous section, we explained that certain decisions needed to be made in order to configure an IQBot to work with a specific system. First, we needed to provide a data access plug-in that can query UniProt data programmatically (including scraping versioning information from the UniProt web pages). UniProt contains a versioned “entry” for each protein. Each entry has a unique identifier, known as the accession number, and in order to access versioned protein entry data we construct a URI using the accession and the revision number. The URI consists of a fixed part (http://www.uniprot.org/uniprot/), followed by the protein accession number and the entry version number. For example, http://www.uniprot.org/uniprot/P42645.txt?version=117. Each protein has at least one entry version, and a new entry version is produced whenever the entry data is changed in a release.

UniProt can provide protein data in various formats. We use the text-based format, in which each line of a protein entry file represents a specific attribute/data type. Lines have an initial code of two letters, giving information about the content of the data on the line. For example, the protein name can be found on a line which has the initial code “DE”. In our code, we extracted the protein name by accessing the protein entry and reading the content of the line with this code.

Second, we must tell the IQBot what kinds of change it should monitor. UniProt provides a range of information about proteins, such as the protein name, publications, organism classification, two- and three-dimensional structure and much more. As mentioned previously, the IQBot will extract changes in data and provide the full history of changes if it is applicable. To keep our experiment achievable, we currently work with just one type of change: curated changes to protein names. Protein names are very important to scientists as key identifying information. However, they are also subject to important changes, as our understanding of what proteins exist and what their roles are changes. For example, if a protein is discovered simultaneously in two different labs, they will very likely upload it twice in UniProt, with different names. When a curator discovers this, and merges the entries, a single new name must be selected for the combined protein. Other scientists who continue to use the old names will cause ambiguities and other quality issues in their data.

The process of comparing changes and assigning the change type remains the same as described in Algorithm 1. Protein names are extracted from consecutive versions of each changed protein entry. The extracted names are compared. If the names are different, then the type of the change is assigned.

Discovering the reasons for the changes is more complex. As mentioned before, they are highly domain specific, and so need to be specified in a plug-in component before they can be worked with. In general, additional information must be sought from the data to determine which kind of reason is behind each change. For example, an important piece of supporting information for name changes are the evidence codes (UniProt, 2015). Evidence codes were introduced into UniProt in September 2015. They are terms from an ontology[3] that describe the strength and type of evidence that exists in support of the protein the entry models. Currently, UniProt is using ten ECO evidence codes: seven for manual curation and three for automatic curation, as shown in Table 1.

Each ECO evidence code refers to a specific type of evidence. For example, if an entry contains the code ECO:0000250, it means that the features of the protein as described in the entry were copied manually from another protein entry that has been found to be highly similar in terms of its sequence to the newly curation protein. This is a weaker form of evidence than code ECO:0000305, which means that the description of the protein in that entry was formulated based on the scientific knowledge of the curator.

By examining the evidence codes in the protein entries before and after a change, we can hypothesise about the reason for the change. For example, if evidence codes grew stronger, then we can guess that the change was due to the arrival of better evidence that overruled the earlier, weaker claim.

For protein entry versions made before September 2015, we cannot count on the ECO evidence code to identify the reason for the change, as it is not provided. As a result, we investigated other information resources to be able to identify the reason for the change. As well as examining the data ourselves, we used an association rule mining tool over the changes to see which other parts of the entries changed alongside protein name change.

[3] The Evidence Code Ontology (ECO): http://www.evidenceontology.org

Table 1. ECO evidence codes used in UniProt entries and their meaning (UniProt, 2015).

Entry Type	ECO	Meaning
Manual	ECO:0000269	provided by experimental evidence.
Manual	ECO:0000303	non-traceable author statement.
Manual	ECO:0000250	sequence similarity evidence.
Manual	ECO:0000312	imported from another database manually.
Manual	ECO:0000305	based on scientific knowledge of the curator.
Manual	ECO:0000255	match to sequence model evidence.
Manual	ECO:0000244	a combination of experimental and computational evidence.
Automatic	ECO:0000256, ECO:0000259	match to sequence model evidence.
Automatic	ECO:0000313	imported information.
Automatic	ECO:0000213	a combination of experimental and computational evidence.

We found five reasons for name changes from observing the collection of 1499 changes in protein names, which were mentioned in the previous section. The reasons, ordered from the most common reason to the least, are:

1. Protein entries are stored in one of two subsets of UniProt: SwissProt or TrEMBL. The TrEMBL database contains protein entries which are automatically curated, while the SwissProt database has the manually curated entries.

   In our sample set of 1499 protein names, we found that the majority of name changes were made when an entry was moved from TrEMBL to SwissProt. This is the point at which a protein entry is first subject to manual curation. Normally, in the first manual curation, the curator reviews all the information in an entry and fixes any errors found, including changing the protein name if required. In our protein names IQBot, we automatically check if the protein name was changed due to first manual curation by comparing the name of the database where the entry is stored, for the version where the change occurred and the previous version. If the databases are different, then it means that the entry was curated manually for the first time.

   This conclusion was backed up by the results from the association rules, which indicated that a name change happened when a protein entry is curated manually for the first time by a curator.

2. There are a number of cases where protein names changed due to merging of protein entries. Merging protein entries means combining several protein entries into one entry. In manual curation, curators are required to follow the database curation guidelines, and one of these guidelines is to merge all entries which have the same gene name or same species into one entry. This results in changes to some protein names (UniProt, 2014). We checked this by comparing the value of the accession number in both versions, the one where the name changed is detected

and the previous one. If the more recent version has more subnames than in the previous entry with the same accession number, then it means a merge was made.

3. Curators manually review the literature on proteins, and when they find new publication related to a protein they may make changes to that protein entry, such as changing the protein name. In our code, we look at the publication section of the protein entry version where the name change is detected. If it differs from the publication section in the previous entry version, and new publications have been added, then it means that protein name change due to new publications in the area.
4. It should be noticed that, in many cases we came across, when some parts of the name were removed, the removed part was added as a flag to the protein entry. After investigating, we realized that this change occurred due to changes in the UniProt naming guidelines[4]. The guidelines do not allow some terminologies to be used in protein names. For example, if the protein name contains the word "Putative", then the word should be removed from the protein name and added to a section in the entry called flags, as this word is no longer considered acceptable in protein names.
5. In rare situations, nothing to support any of the above reasons can be found in the entry data, where name change is found. From reviewing the curation manual for UniProt (UniProt, 2014), it can be said that the curators made changes to protein entries based on their knowledge, expertise and analysis of the publication in the domain. So this would be given as a default reason if none others are suitable.

Experimental Results

In this work, we hypothesised that the IQBot approach can discover defects in data and find corrections for the defects, based on an interpretation of the changes made by data curators. Also, we set out to provide the reasons behind the detected changes. When we used IQBot to monitor UniProt, it detected 1499 changes in protein name from a target set of 249 protein entries.

To test this hypothesis, we accessed a collection of databases which deal with proteins to check whether any of them were using out-of-date protein names. We used DBGET[5], which is an online tool that grants access to a collection of biomedical databases. Using DBGET saves time and effort as we do not need to be familiar with the way of accessing each database individually; DBGET provides the same mechanism to access all provided databases. We implemented a java program that accessed the biomedical databases in DBGET automatically and checked if each database still contains any of the old or updated protein names. We searched in eight databases: KEGG BRITE, GO, KEGG GENES, KEGG DGENES, KEGG ORTHOLOGY, KEGG MGENES, NCBI-Gene and KEGG ENZYME.

Algorithm 2 shows the pseudocode of finding the previous and current protein names in the DBGET databases. For each database, we check the availability of both protein names (current and previous). If the database contains both protein names or only the

[4] UniProt naming guidelines: http://www.uniprot.org/docs/nameprot
[5] DBGET: http://www.genome.jp/dbget/

Algorithm 2 Pseudocode to search different databases in DBGET.

```
Require: currentName : the name the changed protein entry current has
Require: previousName : the name the changed protein entry current had before the
    change
1: results = queryEachDB(previousName)
2: for result in results do
3:     if !result.has(currentName) then
4:         db = result.getDB()
5:         record currentName as being out-of-date in db
6:     end if
7: end for
```

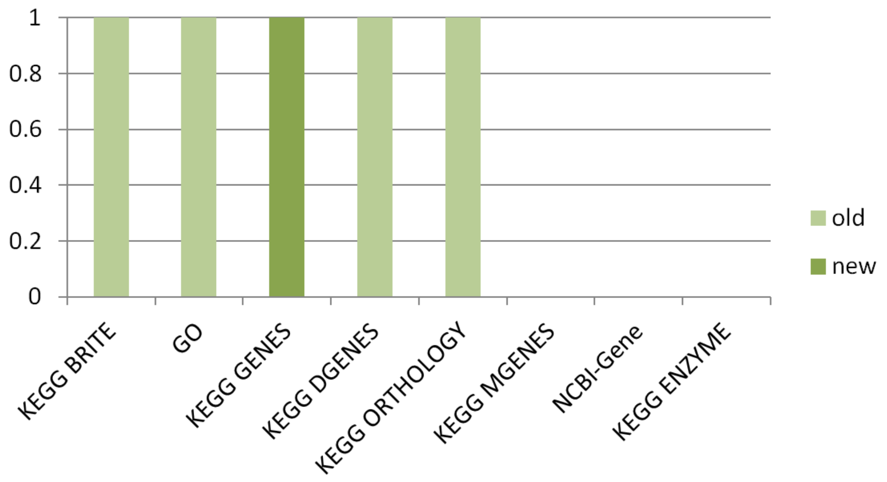

Figure 1. The results of checking the availability of old and new protein names in DBGET

current protein name, then it will be considered as an updated database. However, if the database has only the previous protein name, then the database will be considered as an out-of-date database.

Figure 1 displays the results of the experiment. A value of 1 for a database indicates that the database contains protein names, whether current (new) or previous (old) names. The old refers to the previous protein name and the new refers to the current protein name. A value of 0 means that the database does not contain any of the protein names. Figure 1 shows that only one database, KEGG GENES, contains the most updated protein names. However, four of the databases have out-of-date protein names.

Conclusion and Future Work

In this paper, we proposed a means to amplify the curation efforts made on curated databases to give value to the owners of other databases containing overlapping (but not identical) data. To date, we have tested our IQBot on the well-known (and well-used) UniProt database. The results demonstrated that IQBot can identify changes made by curators and it can find the reason behind the change, and thus the defect. We also located examples of databases containing out-of-date data, which we identified through analysis of the changes made by the UniProt curators.

There are several possible future directions to this work:

ECO evidence code We mentioned previously that the ECO evidence code ontology contain a huge collection of evidence terms and uses by other biomedical databases. This raises the possibility of linking our approach more tightly to the whole ontology of ECO evidence code. By doing this, we aim to give the opportunity to all biomedical databases, which use ECO evidence code, to use the IQBot with minimal changes to the code. They would need only to specify which data to be extracted and how it is accessed. The reason inference plug-in could be reused.

Publishing the results Now that we have shown we can extract defects from curated databases, and find the reasons behind them, it is time to publish the results in a way which make them applicable for other sources to use. Making the results available on the web will help users and owners of databases that are still using out-of-date data to raise the quality of their data by applying the published corrections.

Creating a model for curation Another important next step is to test IQBot against different databases, to see whether the results can still hold. Ideally, we would like to try it with some non-biomedical databases, to allow us to check the generality of our model.

References

Abrams, S., Cruse, P., Strasser, C., Willet, P., Boushey, G., Kochi, J., . . . Rizk-Jackson, A. (2014). Datashare: Empowering researcher data curation. *International Journal of Digital Curation*, *9*(1), 110–118.

Bunt, S. M., Grumbling, G. B., Field, H. I., Marygold, S. J., Brown, N. H., Millburn, G. H., Consortium, F. et al. (2012). Directly e-mailing authors of newly published papers encourages community curation. *Database*, *2012*, bas024.

Embury, S., Jin, B., Sampaio, S. & Eleftheriou, I. (2014). On the feasibility of crawling linked data sets for reusable defect corrections. In *Proceedings of the 1st workshop on linked data quality (ldq2014)* (Vol. 1215).

Kumari, P., Achmiz, S. & Kennedy, O. (2016). Communicating data quality in on-demand curation. *arXiv preprint arXiv:1606.02250.*

Orchard, S., Ammari, M., Aranda, B., Breuza, L., Briganti, L., Broackes-Carter, F., . . . Del-Toro, N. et al. (2013). The mintact project—intact as a common curation platform for 11 molecular interaction databases. *Nucleic acids research*, gkt1115.

Orchard, S., Kerrien, S., Abbani, S., Aranda, B., Bhate, J., Bidwell, S., . . . Cesareni, G. et al. (2012). Protein interaction data curation: The international molecular exchange (imex) consortium. *Nature methods*, *9*(4), 345–350.

Ravagli, C., Pognan, F. & Marc, P. (2016). Ontobrowser: A collaborative tool for curation of ontologies by subject matter experts. *Bioinformatics*, btw579.

UniProt. (2014). Uniprot manual curation sop. http://www.uniprot.org/docs/sop_manual_curation.pdf. [Online; accessed 30 July 2016].

UniProt. (2015). Evidences. http://www.uniprot.org/help/evidences. [Online; accessed 23 May 2016].