On the Feasibility of Crawling Linked Data Sets for Reusable Defect Corrections.

On the Feasibility of Crawling Linked Data Sets for Reusable Defect Corrections

Suzanne M. Embury
School of Computer Science
University of Manchester
Oxford Road, Manchester, UK
suzanne.embury@manchester.ac.uk

Binling Jin
School of Information Science and Engineering
Lanzhou University
Lanzhou, China
bljin@lzu.edu.cn

Sandra Sampaio
School of Computer Science
University of Manchester
Oxford Road, Manchester, UK
sandra.sampaio@manchester.ac.uk

Iliada Eleftheriou
School of Computer Science
University of Manchester
Oxford Road, Manchester, UK
iliada.eleftheriou@manchester.ac.uk

ABSTRACT

Current linked open data standards have encouraged the publication of a large number of data sets on the public Web. While some data providers put a lot of energy and resources into maintaining high quality data, others do not, meaning that the quality of the data in many LOD sources is variable and unpredictable. This makes the construction of novel applications on top of the data more difficult and expensive than it otherwise would be.

However, these same data standards also open up possibilities for new ways of managing information quality (IQ). In this paper, we propose one such approach, the IQ-bot, and present the results of our study of its feasibility. An IQ-bot is a 3rd party component that crawls the Web of data, looking for changes that have been made to data sets, and inferring from them where a correction to a data defect has been made. These corrections can then potentially be made available for application to other databases showing evidence of the presence of the same data defect. In this way, the benefits of the curation effort put into a small number of data sets can be propagated throughout the Web of data.

1. INTRODUCTION

Linked Open Data standards make possible the creation of a range of new applications, built with data sets that were formerly hidden behind web forms or proprietary query interfaces, or else not publicly accessible at all. Unfortunately, the quality of many linked open data sets is extremely variable. Poor quality data can raise the cost of building and maintaining new applications; prohibitively so, in some cases. Or, the problems caused by the poor quality data may only become visible when the application is in active use, and is found to produce incorrect or unhelpful results.

When data consumers must use data that is not fit for purpose, it is common for them to pull a copy of the data set (or a subset of it) to their local space, where the problems can be addressed before it is used. The data may then be translated into a more useful format, or transformed to fit the schema and semantics required for the computational task the consumer has to perform. Data from several sources may be combined, including local data produced by the data consumer's own processes or organisation. And, any defects in the data that the consumer discovers (typically during the attempt to actually use the data) can be corrected.

Ideally, such data consumers will report the defects they find back to the owners of the original source. In practice, this does not always happen (although the process is becoming easier as more sources provide convenient routes for error reporting). Even when errors are reported, the source owners do not always have the resources to be able to check and make the corrections immediately. The result is that the same error may be found and corrected repeatedly by many different data consumers, in isolation from each other's efforts. Few resources exist (beyond promising experiments such as the PatchR repository [11]) to allow data consumers to benefit from the data correction activities of others, except by chance or by a concerted effort on the part of groups of data producers or consumers, the benefits of which may not obviously be worth the associated costs.

There is therefore a need for a mechanism to facilitate sharing of both identified defects and their corrections. Any such mechanism should preserve the freedom of data providers and consumers to structure their data as they wish, and should avoid the imposition of heavy-duty global standards or conventions. In other words, the autonomy of the participating sources should be respected, as should the scarcity of time and attention from the domain experts that are available to find and fix the problems.

In this paper, we propose one possible approach to providing

LDQ 2014, 1st Workshop on Linked Data Quality Sept. 2, 2014, Leipzig, Germany.

such a mechanism: the IQ-bot (short for "Information Quality Bot"). An IQ-bot is a lightweight, 3rd party component that crawls data sets of interest to a community, looking not for data defects *per se*, but for corrections to data. The IQ-bot reports any corrections it finds to a central Data Corrections Server. This server is accessed by data set owners, who can query it to see if any of the corrections found so far might be applicable to their own data. Relevant corrections can then be applied to the local data, to raise its quality level.

The fact that our IQ-bots search for *data corrections* rather than data defects is important. The semi-automated discovery of data defects is a Holy Grail for the IQ community. Many tools have been developed with this goal in mind, the most successful of which are probably the data profiling tools such as TS Discovery[1] and Talend Data Profiler[2]. These tools can help a data owner or consumer to understand the overall landscape of their data, and to identify outliers and other deviations from the "normal" behaviour of the data. However, the identification of true defects from the mass of information provided by the tools typically requires both domain knowledge and an understanding of the state of the world relative to the data at specific points in the past. This, like any task, requiring significant human input, is expensive.

Data corrections, on the other hand, are concrete changes to data, identifiable using entirely automatic means. They form a visible trail in the data, in some cases recorded explicitly in the form of a change (or transaction) log, in other cases encoded implicitly in the difference between versions or snapshots of the data. Some data changes arise from the addition of new data (addressing incompleteness of the data set), while others make changes to existing data (addressing inaccuracies, inconsistencies or a lack of currency in the data, for example). In other words, the changes can be viewed as actions taken by the data owner in response to a perception of poor data quality or the presence of a data defect. The corrections tell us both something about the defect that was identified *and* the action that was decided on as being an appropriate response.

By searching for and storing these corrections, we can gain extra value from the work done by curators and domain experts to locate data defects and remove them. The effort put into curating one resource can be picked up by an IQ-bot, and made available for easy application by the owners of other sources. This acts as a counterpoint to the (currently more common) process of data pollution, whereby data from one source is copied into another, propagating defects from the first source to the second, and onwards to any other dependent sources. IQ-bots give us a mechanism for spreading not defects, but corrections to defects throughout the Web of data. Thus, effort put into raising the quality levels of core data sets can have benefits for a much wider range of data sources, including sources unknown to the curators doing the original defect correction work.

In the remainder of this paper, we present our preliminary study into the feasibility of the IQ-bot approach for finding and applying data corrections on typical linked data sources. We begin by surveying the state of the art in information quality management for linked open data (Section 2). Next, we present an architecture for the IQ-bot approach, showing the core components that are needed and the way they work together (Section 3). Subsequently, we present the results of our two feasibility studies, the first of which explores the extent to which current linked open data sets are versioned or are amenable to snapshot differencing techniques (Section 4) and the second of which considers the changes made to one specific versioned data set, and whether they can be applied to other data sets to improve data quality (Section 5). Finally, we conclude and outline some of the near-term future directions for the work (Section 6).

[1] `www.trilliumsoftware.com/home/products/TSDiscovery.aspx`

[2] `www.talend.com/resource/data-profiler.html`

2. RELATED WORK

Linked Open Data (LOD) is a set of principles and technologies that enable owners of data sets to share their data across the Web. With the constant growth in the number and size of LOD data sources, LOD has given rise to a wealth of freely available, structured information on the Web, which organizations as well as citizens have been taking advantage of by browsing and analysing the open Web of data. LOD is usually represented in RDF (the Resource Description Framework), having the form of an entity-relationship graph in which facts consisting of triples of subjects, predicates and objects are specified. Data sets are typically connected to one another through links between entities, showing where they represent the same or related real world entities.

The growing popularity of LOD has led the information quality community to investigate its quality under a variety of use cases. For example, Kontokostas *et al.* proposed a methodology for quality assessment of linked data, which employs SPARQL query templates that can be automatically instantiated into quality test queries [12]. The test queries are associated with RDF triples and serve a number of purposes, such as assessment of quality of data (including ontologies and vocabulary schemas) considering integrity constraints as well as domain-specific semantics, the creation of libraries of quality test patterns for rapid development of new test cases and a metric for coverage of RDF test cases.

Also addressing the problem of quality of ontologies used with LOD, Abedjan *et al.* identified common ontology misuse patterns by employing frequency analysis and rule mining, and devised an algorithm to suggest individual ontology re-engineering improvements [2]. The aim is to improve the quality of LOD ontologies by overcoming any mismatches between the data and the ontology. Such mismatches arise due to differences between the original purpose for which the ontology was engineered and the unpredictable ways in which new applications utilize it. These mismatches account for the many of the difficulties data consumers face when trying to discover and integrate domain-specific information.

Contributions regarding IQ and data integration in a LOD context include the work of Heise and Naumann, in which a set of data integration operators are proposed and used to extend an existing Web-scale data analysis framework [10]. The proposed operators form an algebra for a declarative query language, Jqal, with which users are able to specify

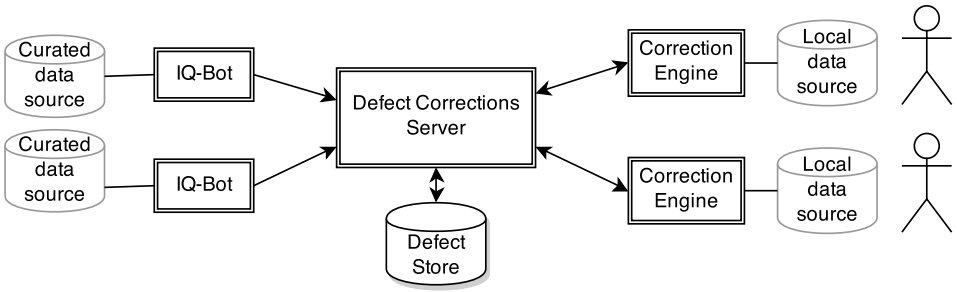

Figure 1: An Example Configuration of the IQ-bot Components

the integration of several, possibly large, data sources into a single, wide and deep data source. The data integration operators implement sophisticated data cleansing techniques, such as record linkage, generating high quality integrated data sets, which facilitate complex queries over the relationships of the different types of entities, yielding the generation of new insights from the data.

Despite the amount of work addressing the use of data dependencies and integrity constraints for error detection and data repairing in recent years (e.g., [4, 7, 8, 14, 9, 5]), these techniques have yet to be thoroughly revisited in terms of their potential applicability to LOD. The work described by Bauckmann *et al.* represents one contribution in this direction. They identify a new case for conditional inclusion dependencies, typically found in LOD data, and propose algorithms for efficiently detecting the presence of such dependencies [3].

Perhaps the most comprehensive contribution to LOD quality management to date is the work of Abedjan *et al.*, in which a Web-based tool for a variety of LOD data management tasks is proposed (ProLOD++) [1]. The tool adapts existing data profiling tasks for relational data to the RDF model. The proposed tool provides a variety of functions based on research in LOD data management, including data profiling tasks for generation of common data profiling statistics (e.g., predicate frequencies, value distributions, etc.); data mining tasks for inverse as well as synonym predicate discovery, identification of dependencies between subjects, predicates and objects; identification of entities via predicate combinations, etc.; and data cleaning tasks for data auto-completion, ontology alignment and fact generation/amendment.

All this work represents progress, but many important aspects of the data quality problem in a LOD context are still unaddressed. To the best of our knowledge, no tool or resource has yet been proposed for identifying corrections in curated LOD sources, to enable the benefits of the work done to maintain high quality in those sources to be propagated further through the Web of data.

3. PROPOSED IQ-BOT ARCHITECTURE

Three different types of component are needed to support crawling the Web of data for data corrections and storing them in a reusable form. Figure 1 illustrates these component types in one typical configuration. In the figure, boxes with a grey outline are pre-existing data sources, while boxes outlined in black show the new components needed to carry out the crawling and correction application processes.

On the leftmost side of the figure, we see a number of pre-existing data sources. These are the versioned, curated data sources that a particular user community wishes to monitor for data corrections[3]. A number of *IQ-bot components* are connected to these sources. The IQ-bots perform regular monitoring of the sources for data updates. For example, an IQ-bot monitoring a database of gene information might discover from an update that the gene formerly known as "FTHFD" is now known as "ALDH1L1"[4]. Exactly what this monitoring entails depends on the sources themselves and the facilities they offer for notifying external agents of changes (or the lack of them). We envisage that multiple IQ-bot components might be active simultaneously in the general case, providing load balancing and also handling different kinds of sources with different change management facilities.

The IQ-bots report any changes they find to a central Defect Corrections Server. This server stores the corrections in its persistent store, as well as potentially classifying and ordering the corrections in order to make them easier to find and apply. In the figure, we show just a single server, although it is possible to imagine scenarios in which multiple servers might exist (where, for example, the different servers store corrections from data in different domains).

Meanwhile, a bioinformatician working with her own collection of gene data (created by combining data generated in her lab with data from species-specific public databases such as SGD[5]) wishes to carry out an important analysis. Before running the lengthy analysis, she first runs her local Correction Engine, to make any improvements to the quality of the data that might have been discovered. The Correction Engine connects to the user's preferred Defect Correction Server, and checks if any new corrections have been discovered since the last update to this source. The Correction Engine then examines the local data sources, looking for opportunities to apply the new corrections to the data. If there are matches, then these are presented to the bioinformatician, who can choose to accept them (apply them to the

[3]In this configuration, the users predefine the list of sources to be monitored. In others, the sources might be identified by some other means, such as by following links in data sets.
[4]`www.uniprot.org/uniprot/Q8R0Y6`
[5]`yeastgenome.org`

data) or reject them as being inappropriate.

The two studies described in the rest of this paper aim to assess the feasibility of providing this kind of facility, in the context of the current linked open data landscape.

4. FEASIBILITY STUDY

In order to determine whether the IQ-bot approach just described is feasible, we undertook a study of a sample of linked data sets. Our aim was to answer the following questions.

- The IQ-bot approach depends crucially on the ability to efficiently identify changes that have been made to data sources. This is made much easier if sources are explicitly versioned by their providers, using software platforms that give details of the changes between versions. We therefore wanted to know: **what software support is available for versioning of linked open data sets?** and **what proportion of linked open data sets currently use this software support?**

- If a data set of interest is not accessible through a version management framework, the IQ-bot approach can still deliver results if previous versions of the data set are made available. That is, instead of providing just the most recent version of the data, with all updates applied, some linked open data providers maintain RDF dumps of previous major versions of their data. In this case, we can identify the changes that have been made in a version by comparing its contents with its most recent predecessor. So, if a source creates new versions relatively frequently, and makes them publicly available for some time afterwards, the IQ-bot approach can be considered feasible if sufficient time is available to perform the snapshot differencing. We therefore wanted to know: **what proportion of LOD sources make their previous versions available, after the data has been updated?** and **what frequency of updates are typical?**

- If versions are maintained for only a very small number of LOD sources, then the IQ-bot approach may not be feasible. In this case, we have another option. We can take regular snapshots of the data set to be monitored, and store the most recent locally for comparison when the next snapshot is taken. By comparing successive pairs of snapshots, we can discover the changes that have been made recently. This option requires only that we can take an RDF dump of the data set (or can produce one through a query or set of queries at a SPARQL endpoint), and that we have sufficient local disk storage to be able to keep copies of the RDF dumps of all sources we are monitoring, between snapshots. This requirement doesn't seem onerous, given current disk storage costs. However, for completeness, we have also attempted to gain answers to the following questions: **what proportion of linked open data sets are available as an RDF dump, or offer a SPARQL endpoint?** and **what sizes are typical for linked open data sets?**

The answers that emerged from our feasibility study are described below.

Versioning Support for LOD. We are beginning to see the emergence of platforms for linked open data that provide explicit support for versioning. The prime example of this at present is Apache Marmotta[6], which offers a versioning module on top of the KiWi data store. When versioning is enabled, a Marmotta application will automatically track changes to RDF data, at the transaction level, and allow specific versions of the data set to be queried. It also supports the creation of snapshots of the data, which can be referred to and queried directly, too.

This kind of facility is exactly what we need to implement IQ-bot-style correction crawling efficiently. Unfortunately, we couldn't find any publicly accessible versioned data sources that used Marmotta. The main RDF triple stores (we looked at Jena's TDB[7], Virtuoso[8] and Sesame[9]) offer support for transactions but not versions *per se*. Some proposals for handling versioning of RDF data can be found in the literature (e.g. the work of Van de Sompel *et al.* [13]). But, these have not so far resulted in widespread use of versioning mechanisms for the management of linked open data. This is a development that will presumably come when the field has matured to the level where normal data management facilities are available as standard for linked open data, just as they are for current transaction-based information systems.

Informal Version Management for LOD. Since we found little evidence that linked data providers are making use of versioning tools, the feasibility of the IQ-bot approach requires that other, more informal approaches to version management be in widespread use. In order to assess how far this might be the case, we examined a list of 902 data sets, provided by the *Datahub.io* website. Datahub.io is a data management platform where users can publish, register and access LOD from organisations around the world and across a range of application domains. It therefore provides access to a good cross-section of the range of LOD sources.

To quickly gain a picture of how many of these linked open data sets were versioned to some extent, we used a script to query the description pages on Datahub.io, to find out how many data sets were using the "versioned" tag. We found that 253 data sets described themselves as being versioned, constituting some 28% of the full collection. This sounds promising, as far as the IQ-bot approach goes. Unfortunately (though predictably), the accuracy of this tag seems to be in doubt. We found examples of data sets not tagged as versioned in Datahub.io that had details of recent changes, and other versioning information, on their home Web pages. Similarly, many of the data sets that described themselves as being versioned gave little evidence of this on inspection. Around 120 of the data sets tagged as versioned listed their version number as "0.1" or "1" or some similar string suggesting that versioning was intended, rather than actually achieved by the data owners. Around 50 data sets gave a version number greater than 1 (or 0.1), indicating that at least two versions of the data have been in existence at some point, while the others gave date of last update as their version ID, from which we could conclude little. Even

[6] `marmotta.apache.org`
[7] `jena.apache.org`
[8] `virtuoso.openlinksw.com`
[9] `www.openrdf.org`

as an under-estimate, 50 out of 900 is perhaps a rather small proportion, but still leaves room for IQ-bots to be of use if these 50 include some major curated data sets in varying domains. (They do; for example, WordNet, DBpedia live, Linked Life Data, and data sets from Bibliotheque Nationale de France are included).

Snapshot Comparisons for LOD. Many more data sets become eligible for IQ-bot-style crawling if we can record snapshots of the source contents at regular intervals, and work out the differences between them. We again used a script to query the Datahub.io index of data sets. The tags indicate that just 27 out of the full set of 902 data sets offered an RDF dump of their data. This seems very low, calling the accuracy of this tag into question. We did a manual search of 5 of the sources, and found that 4 out of 5 offered an RDF dump, but on their own Website and not through Datahub.io. Data sizes of these dumps range from 17,000 triples to greater than 1 billion triples. Clearly, the storage of many RDF dumps from the larger sources would have to be justified by significant data improvement resulting from the data corrections that could be detected and reused.

The tags on Datahub.io suggest that 510 of the data sets offer a SPARQL endpoint through datahub, which looks promising as a last ditch route to obtaining snapshots. However, the (anecdotal) experience of our colleagues working with linked open data is that the proportion of data sets with live, usable SPARQL endpoints is likely to be much lower than this.

5. DATA CORRECTIONS IN UNIPROT

Our general feasibility study showed that there are sufficient access routes to change information for linked open data sets to make IQ-bots a realistic option in certain domains, even if some work has to be put in (for example, to allow snapshot comparisons). We further wanted to assess whether we could identify actual data corrections (and the defects that lie behind them) from a real data source, and whether the corrections we found could be applied to other data sources. To achieve this, we needed a versioned, well-curated, widely-used data source. We chose the well known UniProt database[10] for this second feasibility study.

UniProt is a comprehensive, heavily curated database of information on protein sequence, structure and function. The database aims to document the set of proteins that have been discovered so far, plus our current understanding of their form and function within living organisms, and the evidence (experimental or otherwise) that supports this interpretation. It is a major global resource for scientists working in molecular biology and cognate disciplines.

Information in UniProt comes from two main sources. A team of expert curators works to extract high-quality information from the scientific literature, and to associate this with the relevant protein records (called "entries" in UniProt terminology). Though well resourced, this team can cover only a small fraction of the proteins that are stored in the database, with new proteins being discovered frequently every year. Therefore, a second source of information is utilised: automated analyses of the literature and the data in UniProt make predictions about the features of un-curated proteins. This data is also added to the database, but is annotated with evidence codes to indicate to users that it comes with a weaker supporting evidence base than the information entered by the expert curators.

The UniProt team produces a new release of the database every four weeks. We examined a single release, to see if we could identify defect corrections from the changes made in that release. UniProt was a good choice for this study because of the high frequency of changes made by the curators, and because it provides several ways to access the details of the changes that have been made in each release:

- The UniProt Web search form allows comparison of arbitrary versions of a particular entity, showing the results as a difference between two textual representations of the entry from the selected versions.
- A Java API for accessing UniProt remotely allows us to query for the entries that have changed in a particular release. (We then have to compare the updated entries for ourselves, before we can get information about the specific changes that have been made.)
- Although access to these versioning facilities is not currently available for RDF/SPARQL users, the UniProt database is made available as an RDF dump, leaving open the possibility of storing the dump after each release, for snapshot comparison purposes.

Since for our feasibility study, we planned to do the analysis by hand, we chose to use the search form for extracting a text-based representation of the changes. We examined changes made in the release covering 1st March 2014 to 24th March 2014. During this time, changes were made to 5,139 protein entries in UniProt. This is typical of the scale of changes that occur in each release. Of these changed entries, we looked at 14 in detail to extract data corrections[11]. Most of the changed entries had been updated in several ways, so the actual number of detailed changes we examined was 134. By "change" here we mean a small conceptual change, such as adding a new feature to a protein or deleting a keyword from a set of keywords. Since we were working with the textual comparison interface, without full knowledge of the underlying database schema, it was not possible for us to work out in detail the exact number of attribute value changes, additions and deletions. Table 1 summarises the results.

Space precludes us from giving our detailed analysis of each of the entries. Instead, we will present some examples of the changes we examined, and our classification of them according to the underlying defect, for the different IQ dimensions covered.

- *Completeness:* several of the entries were changed in order to increase the completeness of the information stored. As our collective understanding of the structure and function of proteins grows, incompleteness

[10] `uniprot.org`

[11] This number was chosen based on the amount of time we had available to research each changed entry, rather than having any significance in itself.

Entry ID	Versions Compared	No. of Changes	IQ Dimensions Covered	No. of Changes Not Interpretable as Defect Corrections
P62258	119 → 120	11	Consistency, currency, completeness	0
P31947	148 → 149	8	Consistency, currency, completeness	0
Q02152	125 → 126	3	Currency	1
P00350	122 → 123	1	Accuracy	0
P12023	177 → 178	5	Completeness, currency	0
P54645	133 → 134	22	Currency	0
P14164	125 → 126	3	Consistency	0
P31414	115 → 116	67	Consistency, currency	0
P98057	94 → 95	2	Precision	0
P0CI58	9 → 10	1	Syntactic accuracy	0
D9U298	12 → 13	1	Syntactic accuracy	0
Q9LFN6	91 → 92	7	Consistency, currency, completeness	0
P32169	112 → 113	1	Accuracy	0
P0C6Y0	49 → 50	2	Completeness, currency	0

Table 1: Summary of the Entries Examined from the March 2014 UniProt Release

can arise in our scientific databases, unless work is done to repair it. This is a major task for the expert curators for UniProt, who examine the literature, and amend the data to include new results and interpretations appearing in credible research sources. An example of this can be seen in the changes to entry Q9LFN6. Here, a new publication has been added (indicating a new source of evidence about the protein), and new features have been added to the entry as a result. The old descriptions of the function and subcellular structure of this protein were annotated with the phrase "By Similarity". This means that they were added to the entry not because of strong experimental evidence for this particular protein, but because strong experimental evidence linked these terms to a protein that is structurally *similar* to this one. The new publication apparently provides information and evidence directly about this protein. The old features acquired "by similarity" were therefore replaced with the new information, for which stronger evidence exists. This change increases both the completeness and also the credibility of the entry.

Another source of new data for UniProt comes from the results of automated analyses. New results can be generated when better algorithms are implemented, or when existing algorithms are run over new (more complete) data sets. This kind of completeness correction can be seen in entry P31414. It seems likely that all the 67 changes made to this entry in this record arose because new information was added to a protein to which P31414 is similar. Prior to this release, this entry had a small set of annotations postulating the existence of various helical structures within the protein chain. The evidence codes indicated that these annotations had come from an automated analysis tool. In this release, these speculative annotations are replaced by a host of more detailed and precise annotations coming from another protein "by similarity". Presumably, new (strongly supported) information had become available for this similar protein, and was therefore copied over in the entry for P31414.

This particular completeness correction is a complex one; it has elements of consistency, currency and precision. (We have noted elsewhere the fact that real data defects often combine elements of several IQ dimensions, rather than being cleanly classifiable within the scope of just one [6].) Other, more conventional, forms of completeness correction can be seen in the set of sample entries. Of particular note in the context of this paper are the additions of new links to other datasets. Entries Q02152, P12023 and P0C6Y0 all had additional links to other data sets added in this release.

- *Currency:* both entries P62258 and P31947 have a change to an annotation placed on a link to another database. In this case, the link is to a record in BioGrid[12], a database of protein and gene interactions. The value that annotates the link is a number which seems to correspond to the number of interactions currently present in BioGrid for the protein that the entry represents. If a recent new release of BioGrid had more interactions for this protein, then the count of the interactions stored in UniProt would need to increase, to maintain consistency with BioGrid, and currency of the number of interactions recorded. This appears to be what happened to both these entries in this release.

- *Accuracy:* the entry with accession number P322169 was involved in just one change: an update to correct the end page number on one of the publications associated with the entry. The page numbers for this paper were changed from "1454–1474" to "1454–1462". It is not clear whether the error being corrected here arose during data entry into UniProt, or whether erroneous data was copied from another system.

- *Syntactic accuracy:* the changes to entries P0CI58 and D9U298 both involve the correction of a typo. Both entries contain a link to a specific record in the InterPro database[13] with accession number IPR002061. InterPro is a database that uses predictive models to assign proteins to protein families, based on their sequence and structure, and information in other databases. As well as pointing to the relevant InterPro record, entries

[12] `thebiogrid.org`
[13] `www.ebi.ac.uk/interpro`

in UniProt appear also to contain a copy of the family name from InterPro.

In this case, a typo was present in the family name. Originally, the family name was given as the string "Scorpion_toxinL/defesin", but this is not correct. The last word of the family name should be "defensin". Presumably, this error was originally present in InterPro, but has since been corrected there, and this change is the propagation of that correction to UniProt. (We cannot easily check this, as InterPro does not appear to provide access to its recent versions.)

- *Precision:* An example of a correction where information was made more precise can be found in entry P98057, where two terms describing the function of the protein are replaced by the term that is their common child in the ontology (in this case, the GO ontology of structural and functional terms for genes and proteins[14]). The version of this entry prior to this data release was annotated with the following two terms:

  – ion transmembrane transport
  – proton transport

  In the version of the entry that is part of this March 2014 release, however, these two terms were removed and replaced with:

  – hydrogen ion transmembrane transport

  which is a child of both the deleted terms in the GO ontology. This change, then, appears to be a refinement of the interpretation of the function of this protein, to use a more specific (and therefore more precise) term.

Our experience in looking at these changes was encouraging. With some basic domain knowledge, and some checks of related information in other databases, we were able to form a set of hypotheses about the kinds of defect being corrected in the case of all but one of the changes. This hard-to-classify change was an update to an evidence code assigned to an annotation. Unlike the other changes to evidence codes we observed, this involved replacing a (slightly) stronger evidence code (whose source was given as the UniProtKB itself) with a weaker evidence code (with its source labelled as being the Ensembl database). We could not form any good hypotheses as to why this stronger evidence code was being replaced by a weaker one, and therefore had difficulty in categorising this correction under one of the defect types. However, it is entirely possible that this failure is due to gaps in our knowledge of this domain, and of the specific circumstances of this change.

We next set about looking for other databases (not just copies of UniProt) where some of these corrections could be applied, to improve the data quality. Since we were performing this process manually, we were limited to shallow Web searches using Google, and (where available) individual deep Web searches using keyword based search forms on the resources themselves. Nonetheless, we were able to identify the same defect present in other databases for several of the changes. Here, we list a few representative examples:

- *The replacement of the two GO terms with their common child* (from entry P98057). We found a couple of databases which had the two deleted GO terms from this change, but not the child term. One example is from ZFin, the database of gene information about the Zebra Fish species[15]. The two deleted terms appear in the set of "biological process" annotations for the ATP6V1AA gene. Of course, there may be good reasons why the child term is not used in the case of this gene; an expert decision would need to be made before the correction could be safely applied. But, the IQ-bot can flag the existence of the possibility to the expert, and assist in making and documenting the change if the expert decides to apply it. We also found the pattern of terms relevant to this correction in BioCyc[16], on the record for the gene mrpE (accession number: BSU31640), and in SGD[17], on the record for the gene YGR020C.

- *The correction for the "Scorpion_toxinL/defesin" typo* (from entries P0CI58 and D9U298). We were able to locate a database containing this typo: the LAMP database of anti-microbial peptides, managed by Fudan University in China. The relevant record could be found at the following link, at the time of writing:

  `biotechlab.fudan.edu.cn/database/lamp/detail.php?id=L01A000663`

- *The addition of a new publication* (to entry Q9LFN6). We were not able to locate a data source that has exactly the same set of publications as the UniProt version of this entry before the March release. However, we did locate databases with records for this protein where this publication was missing. This was the case for both the Gramene database[18], which recorded no publications at all for the protein (despite having schema elements for the recording of publications), and the NCBI database[19] (which records the full set of publications for the gene, but only one publication for the corresponding protein).

From the above, we can see that we were able to view almost all the changes in the examined entries as defect corrections, and that we were able to infer something about the form and kind of the defect that is being corrected by the change. With a little effort, we were also able to locate other datasets where the corrections might have been able to point out problems to the data owners. The next step is to undertake a more systematic study, using a prototype IQ-bot, to discover whether more places where the corrections could be applied can be discovered by chasing links in the linked data sets.

6. CONCLUSIONS

From our first feasibility study, we showed that some access to change information can be gathered from a significant proportion of linked open data sets, even if specific software support for working with versions is not at present

[14] `geneontology.org`

[15] `zfin.org`
[16] `biocyc.org`
[17] `yeastgenome.org`
[18] `ensembl.gramene.org`
[19] `www.ncbi.nlm.nih.gov/protein/238481238`

widely adopted. The usefulness of the (somewhat simple) version management facilities provided by UniProt (behind the scenes, when viewed from a linked data perspective) in carrying out our second feasibility study illustrates the value that could accrue if more use is made of tools such as Apache Marmotta's versioning module.

A second feasibility study, focussing on UniProt, also showed that differences between versions of well curated sources can be a rich source of information on the data defects discovered in those sources by the curation processes (both manual and automatic), as well as providing details of how to deal with those defects. Defects from all the basic IQ dimensions were discovered in our small study of a tiny part of just one release, which bodes well for the usefulness of the technique as a whole, when applied to entire releases over a period of time. In addition, we were able to find related databases that also contained some examples of the defects these corrections were intended to address.

We are in the process of implementing our first IQ-bot, plus a Corrections Server to record the corrections/defects it finds and a simple Correction Engine to allow the recorded corrections to be applied to a locally stored data set. Our first IQ-bot monitors the UniProt database for changes, using its embedded versioning support that allows us to request a list of changed entries between specific versions programmatically.

A number of important research questions remain to be answered before a full IQ-bot system can be deployed for actual use, and before the value of the approach proposed here for crawling for data corrections can be assessed. How far is snapshot comparison a realistic approach to obtaining update information from real scale data sets? Can all changes to data sets be seen as data corrections (that is, as changes that improve the data quality in some way)? If not, how can we distinguish changes which do improve data quality from those that don't? What information must be stored alongside each data correction, to enable it to be reused in a wide range of contexts, and also to prevent corrections from continually being suggested in contexts in which they are not suitable? What information must be stored to allow the data owner to make good decisions about whether to apply a correction to a data set (or record) or not? Can we find places where the corrections might be applied by following the links in the data, or are more general deep Web searches required? In our current and future work, we are exploring these and other questions relating to the concept of IQ-bots.

7. REFERENCES

[1] Z. Abedjan, T. Gruetze, A. Jentzsch, and F. Naumann. Profiling and Mining RDF Data with ProLOD++ . In *Proceedings of the IEEE International Conference on Data Engineering (ICDE), Demo*, Chicago, IL, 2014.

[2] Z. Abedjan, J. Lorey, and F. Naumann. Reconciling ontologies and the web of data. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1532–1536, New York, NY, USA, 2012. ACM.

[3] J. Bauckmann, Z. Abedjan, U. Leser, H. Müller, and F. Naumann. Discovering conditional inclusion dependencies. In X. Wenchen, G. Lebanon, H. Wang, and M.J. Zaki, editors, *CIKM*, pages 2094–2098. ACM, 2012.

[4] G. Beskales, I.F. Ilyas, L. Golab, and A. Galiullin. Sampling from repairs of conditional functional dependency violations. *VLDB Journal*, 23(1):103–128, 2014.

[5] M. Dallachiesa, A. Ebaid, A. Eldawy, A.K. Elmagarmid, I.F. Ilyas, M. Ouzzani, and T. Nan. Nadeef: a commodity data cleaning system. In K.A. Ross, D. Srivastava, and D. Papadias, editors, *Proceedings of 2013 SIGMOD Conference*, pages 541–552. ACM, 2013.

[6] S.M. Embury and P.Missier. Forget Dimensions: Define Your Information Quality Using Quality View Patterns. In L. Floridi and P.Illari, editors, *The Philosophy of Information Quality*, volume 358 of *The Synthese Library*, pages 25–41. Springer, 2014.

[7] W. Fan, J. Li, S. Ma, T. Nan, and W. Yu. Towards certain fixes with editing rules and master data. *PVLDB*, 3(1):173–184, 2010.

[8] W. Fan, J. Li, S. Ma, T. Nan, and W. Yu. Interaction between record matching and data repairing. In T.K. Sellis, R.J. Miller, A. Kementsietsidis, and Y. Velegrakis, editors, *Proceedings of 2011 SIGMOD Conference*, pages 469–480. ACM, 2011.

[9] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The llunatic data-cleaning framework. *PVLDB*, 6(9):625–636, 2013.

[10] A. Heise and F. Naumann. Integrating open government data with stratosphere for more transparency. *Journal of Web Semantics*, 14:45–56, 2012.

[11] M. Knuth, J. Hercher, and H. Sack. Collaboratively Patching Linked Data . In *Proceedings of 2nd International Workshop on Usage Analysis and the Web of Data (USEWOD)* , pages 33–40, Lyon, France, apr 2012. ACM.

[12] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 747–758, Geneva, Switzerland, 2014.

[13] H. Van de Sompel et al. An HTTP-Based Versioning Mechanism for Linked Data. In C. Bizer et al., editors, *Proceedings of the WWW2010 Workshop on Linked Data on the Web (LDOW'10)*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[14] M. Yakout, L. Berti-Equille, and A.K. Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of 2013 SIGMOD Conference*, pages 553–564, 2013.