Biomedical Ontology Evolution in the EMBL-EBI Ontology Lookup Service

Olga Vrousgou
European Bioinformatics Institute (EMBL-EBI),
European Molecular Biology Laboratory,
Cambridge
United Kingdom
olgavrou@ebi.ac.uk

Tony Burdett
European Bioinformatics Institute (EMBL-EBI),
European Molecular Biology Laboratory,
Cambridge
United Kingdom
tburdett@ebi.ac.uk

Helen Parkinson
European Bioinformatics Institute (EMBL-EBI),
European Molecular Biology Laboratory,
Cambridge
United Kingdom
parkinson@ebi.ac.uk

Simon Jupp
European Bioinformatics Institute (EMBL-EBI),
European Molecular Biology Laboratory,
Cambridge
United Kingdom
jupp@ebi.ac.uk

ABSTRACT

As ontologies are playing an increasingly important role in data annotation on today's Semantic Web, there is a need to observe their constant evolution. At EMBL-EBI ontology preservation and curation is of growing value, and new tools are being developed to help undertake these tasks. One of these tools is the Ontology Lookup Service which holds 140 public bio-medical ontologies that are updated when new ontology version become available. To track changes in these ontologies we have utilized DIACHRON, a platform for tracking the evolution of RDF documents. With DIACHRON in OLS we can track and store the changes between ontology releases and view the differences through a graphical interface.

CCS Concepts

• Web Ontology Language (OWL) • Ontologies

Keywords

Ontologies, OWL, Data evolution

1. INTRODUCTION

Data integration is intrinsic to how modern research is undertaken in areas such as genomics, drug development and personalised medicine. To better enable this integration a large number of biomedical ontologies have been developed by the community to provide common metadata vocabularies. There are now several hundred biomedical ontologies in widespread use that describe concepts such as genes, molecules, drugs and diseases. This amounts to millions of terms that are interconnected via relationships that naturally form a graph of biological knowledge.

The European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI) is a major provider of bioinformatics services and is involved in the preservation of data and the curation of data so that it can be served back to the community in novel and useful ways. A large part of the curation and added value offered by EMBL-EBI is via the semantic annotation of data with ontologies. Ontologies can make data more interoperable and can be used to enhance search[1] and visualization[2] applications over data.

One of the challenges in working with highly interconnected data is dealing with elements that change. Biomedical ontologies aim to represent the state of knowledge in biology, but this is constantly changing as new biology is discovered. As ontologies develop to describe all domains of biology they must be regularly updated with new terms or refinement of existing terms to stay relevant with the data they describe. Tracking the changes within the ontology alongside the changes in the data is especially challenging. Typically the ontologies are developed independently of the data they annotate, so when ontologies change, it can be difficult to propagate that change to all the datasets that were described with those ontologies.

The DIACHRON project[1] has been developing technology for monitoring the evolution of data on the Web. Dataset versions can be archived in the DIACHRON system and there are components for detecting and reporting on changes in the data [3]. The DIACHRON platform is able to archive and monitor changes in data expressed in the W3C Resource Description Framework[2] (RDF) [4], thus making it suitable for archiving ontologies expressed in the W3C Web Ontology Language[3] (OWL) that can be serialised in RDF.

[1] http://www.diachron-fp7.eu

[2] http://www.w3.org/RDF/

[3] http://www.w3.org/OWL/

The EMBL-EBI has recently developed a new version of the Ontology Lookup Service[4] (OLS) that includes DIACHRON functionality for archiving and executing change detection over ontologies. In this paper we present the design and implementation of the DIACHRON functionality adapted for the OLS system and evaluate the DIACHRON functionality for monitoring ontology evolution.

2. REQUIREMENTS

OLS provides access to over 140 bio-medical ontologies. Ontology documents typically reside on the Web and the OLS system monitors these documents for updated versions, and automatically loads new ontologies into the system when a new version is released. OLS provides services for searching and visualising these ontologies and also provides an API for programmatic access. Many database curation systems use the OLS to find terms that are used in annotating biological data. This process introduces a dependency on ontologies from the annotated data, so any changes in the ontologies can have downstream consequences on the data and any services built over those data.

OLS requires the ability to track terms seen in ontologies over time to better assist applications that rely on these ontologies. There are a large number of changes that one might consider tracking within an ontology. One of the most important aspects of change relating to ontologies is the creation, deletion and editing of term (or class level) information. Being able to track additions to an ontology is also a useful metric of ontology activity. Ontologies that haven't been updated for a long time may suggest the ontology is no longer actively maintained. Ontology best practices[5] encourage developers to a avoid deleting ontology classes, and rather use a deprecation or obsoletion strategy so that terms remain in the ontology or at a minimum the term URIs remain resolvable. This is import for applications that use third party ontologies for data annotation and rely on those terms resolving on the Web. In practice there are still many reasons why an ontology term may get deleted, or a URI may be refactored, so old URIs are frequently no longer accessible.

Deletions in OWL, when viewed at the RDF triple level, are typically associated with a kind of edit. For example, moving a class in the ontology class hierarchy would typically be detected at the RDF triple level as a removal of a triple with an rdfs:subClassOf predicate with the addition of a new triple with the same subject and predicate, but a different object. The ability to detect such changes in an ontology is useful as many edits involving the rdfs:subClassOf predicate may suggest an ontology is undergoing a major rearrangement and as such could have a potential impact on any application that makes an assumption about the structure of the hierarchy. There are also other types of non-logical edits, such as to term metadata that may include the editing of a term label or addition of new synonyms or definitions that might indicate a refinement or change to the interpretation of a term.

Table 1 outlines the major ontology changes that the OLS application is concerned with detecting. These changes are restricted to those that can be expressed at the level of RDF triples.

Table 1. Abstracted ontology changes detected by DIACHRON

Ontology change	Change triplet
Addition of new class	Addition of ?x rdf:type owl:Class
Removal of class	Removal of ?x rdf:type owl:Class
Edit label	Addition of ?x <label property> <OWL literal> Removal of ?x <label property> <OWL literal>
Edit synonym	Addition of ?x <synonym property> <OWL literal> Removal of ?x <synonym property> <OWL literal>
Edit definition	Addition of ?x <definition property> <OWL literal> Removal of ?x <definition property> <OWL literal>
Class move	Addition of ?x rdfs:subClassOf ?owlClass Removal of ?x rdfs:subClassOf ?owlClass
Class obsoletion	Addition of ?x owl:deprecated “true”

2.1 Diachron services

The DIACHRON platform is comprised of several components that are accessible via Web services. The OLS application is currently utilizing services from three of the components, namely, the DIACHRON archive, the change detection, and the integration layer.

- The archive service is responsible for converting the ontology versions represented in OWL to the DIACHRON RDF model. Once converted, the “diachronised” RDF can be uploaded into the archive via the archiving API.
- The change detection service is responsible for the calculation of changes between two ontology releases. There are two types of changes, simple and complex. Simple changes capture all the additions and deletions of all terms in an ontology and are predefined. Complex changes are user defined changes based on specified collections of simple changes.

[4] http://www.ebi.ac.uk/ols/beta/

- The integration layer is designed to provide an abstraction over the underlying services and handle security and mediation of services via a single point of entry to the DIACHRON platform.

3. METHOD

Tracking ontology evolution is the primary use case for DIACHRON within EMBL-EBI. OLS currently deals with tracking external ontologies for changes and indexing any new releases. OLS provides a REST API to the available ontologies that includes information about when an ontology was last updated. We have developed a Java application that crawls the OLS API for each new ontology release, retrieves the latest ontology version and archives it into the DIACHRON platform. As new versions become available via the OLS API, the DIACHRON platform will archive the newer version and run change detection between versions.

For a delta of changes to be produced between the new release and the old version of an ontology, there is a need to always keep the latest versions archived in a uniform way. In order to adapt DIACHRON to be used in OLS, we need to configure the change types in Table 1 to reflect how certain features are implemented in a particular ontology. For instance, most ontologies have a notion of synonyms, but the predicates for synonym vary between ontologies. We would like to use DIACRHON to detect changes, such as “synonym edit” at a conceptual level, without worrying about implementation details. The deltas between ontologies need to be stored and presented in a user friendly way in the OLS application.

The new OLS platform provides an extensive list of API endpoints that provides an easy way for developers to access its underlying data. Using this API we are able to retrieve information about the ontologies that are stored in OLS such as their location where the ontologies will be downloaded from, the latest version that was indexed into OLS, and information such as the predicate used for label, synonyms and definitions.

DIACHRON provides an ontology for describing changes according to the underlying DIACRHON model. There is also an API for creating change types for a given dataset that is exposed as a Web service in DIACHRON. The OLS crawler reads an ontology configuration from the OLS API, creates a new dataset and specifies DIACHRON changes using the DIACHRON service API. The flow for adding OLS data to DIACHRON is as follows (also depicted in Figure 1).

For each ontology in OLS:

1. From the OLS API get the latest version that has been indexed in OLS.
2. Get the latest version that has been archived in the archiver.
3. If it is the first time that this ontology will be archived create a DIACHRON dataset id. We need to know the dataset id before we can upload the “diachronised” dataset.
4. If it is the first time this ontology has been archived, define the ontology’s complex changes and store them through the changes API.
5. If there is a new version of the ontology, download it from the file location that is provided by the OLS API
6. The ontology is downloaded in native OWL or OBO format then submitted to the DIACHRON archive service.
7. If this is a new version of an ontology that has been previous archived, change detection between the new and old versions will be executed using the changes API. The results of the change detection are also stored in the archive and made available via the integration layer.

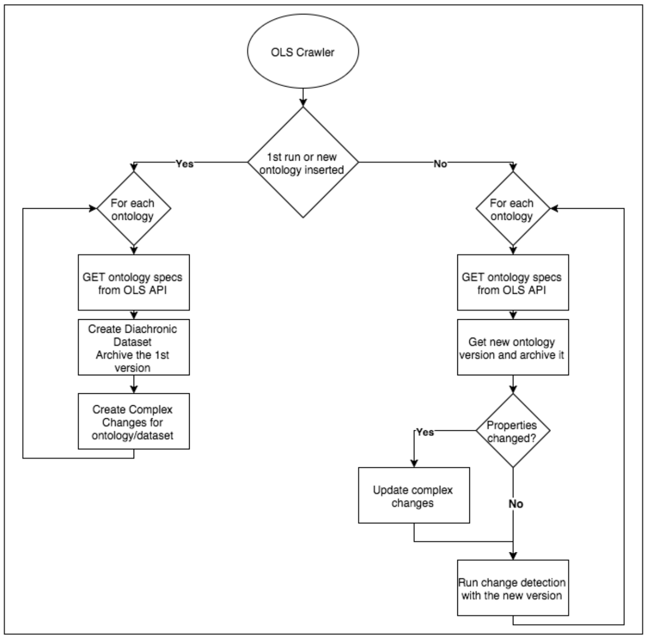

OLS Crawler
1st run or new ontology inserted
Yes
No
For each ontology
GET ontology specs from OLS API
Create Diachronic Dataset Archive the 1st version
Create Complex Changes for ontology/dataset
For each ontology
GET ontology specs from OLS API
Get new ontology version and archive it
Properties changed?
Yes
Update complex changes
No
Run change detection with the new version

Figure 1. Program flow of the OLS crawler. The crawler is scheduled to run every night, pushing new ontologies to DIACRHON, archiving new ontology releases, and running the change detection service for updated ontologies.

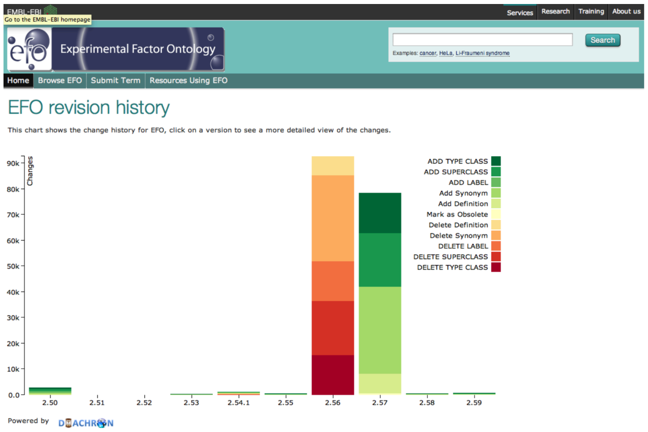
As mentioned above, there is a need to create different complex changes for each ontology. Many ontologies use different properties to describe a term. This leads to the need to define a set of complex changes for each ontology, based on the properties defined in the OLS API. It is also essential to update the complex changes if a term property alters from one version to another. The main challenge we face here is that a complex change may not only have different properties between ontologies, but may also have an entirely different implementation. For example, when a term is marked as obsolete in EFO, it is moved to be a subclass of the *“ObsoleteClass”* class. In other ontologies, such as the Gene Ontology, they use the owl:deprecated property to indicate term obsoletion. We have defined additional configuration into our OLS crawling application that specifies the obsolete class strategy used by a particular ontology. This requires a priori knowledge of the strategy used by an ontology, and requires maintenance of this additional configuration to ensure it remains in sync with each ontology.

4. RESULTS

Changes detected between ontologies are stored in the DIACHRON platform according to the DIACHRON changes ontology schema[6]. These changes can be queried directly with SPARQL or by a REST API endpoint that is provided by DIACHRON to access a JSON representation of the changes. This changes API is used by OLS to visualize DIACHRON changes.

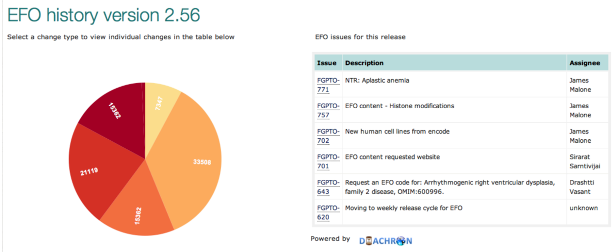
Figure 2 shows an overview of changes in the Experimental Factor Ontology over a 12 month period (EFO version 2.50 through 2.59). This visualisation gives a broad overview of changes in the ontology that is color coded according to the change type defined in Table 1. This provides a useful visual history of the evolution of the EFO ontology. We can easily see that in general the ontology changes very little between releases, with only a handful of new terms added on each release. However, there are two very clear spikes in 2.56 and 2.57, where a large number of deletions were followed in the next version by a large number of additions. This spike exposes a problem with the 2.56 release where many terms disappeared and was quickly fixed for the 2.57 release. The ability to see such a dramatic change is a possible indicator that 2.56 should not be used and any application relying on it should update their EFO version.

Figure 3 shows how we can use the visualization to drill down into specific changes for a given release. We have also connected the visualization component to the JIRA tracking system for the EFO ontology so changes can be viewed alongside tickets that were closed on that release. This feature allows users to see changes in the context of the work that was performed for that release.



This chart shows the change history for EFO, click on a version to see a more detailed view of the changes.

Figure 2. EFO changes from release 2.50 - 2.59.



Select a change type to view individual changes in the table below

EFO issues for this release

Issue	Description	Assignee
FGPTO-771	NTR: Aplastic anemia	James Malone
FGPTO-757	EFO content - Histone modifications	James Malone
FGPTO-702	New human cell lines from encode	James Malone
FGPTO-701	EFO content requested website	Sirarat Sarntivijai
FGPTO-643	Request an EFO code for: Arrhythmogenic right ventricular dysplasia, family 2 disease, OMIM:600996.	Drashtti Vasant
FGPTO-620	Moving to weekly release cycle for EFO	unknown

Figure 3. Reported changes from EFO release 2.56.

5. Conclusion

We have presented an update to the EMBL-EBI Ontology Lookup Service to include DIACHRON base change tracking of ontologies. The DIACHRON platform uses the OLS API to detect when new ontology versions have been indexed and archives new version in the DIACHRON archive. DIACHRON changes are configured according to properties outlined in the OLS ontology configuration, so that a range of abstracted common changes can be detected between all of the ontologies in OLS. We have developed user interface components to assist users in searching and browsing ontology changes.

This work presents a pragmatic and scalable approach to change detection in ontologies. Computing true change detection in OWL ontologies is still an area of active research. Ontology change detection at the level of RDF triples alone is limited to fairly trivial changes in the ontology. However, we have illustrated that this is sufficient to highlight some key aspects of change within ontologies, and this may be sufficient for the users of these ontologies. We are not attempting to report on all ontological changes, but instead require a system that tracks terms and basic term metadata over time. In this scenario DIACHRON provides us with a convenient platform for tracking ontology evolution at this level.

The addition of DIACHRON functionality to OLS provides the OLS user community a novel mechanism to access ontology changes via the DIACHRON API. As the amount of biological data annotated to ontology terms continues to grow, it is vital that tools are in place to assist database developers and curators to track the evolution of these vocabularies. The DIACHRON functionality added to OLS now provides a platform for monitoring these changes and can be exploited in downstream services relating to data concurrency, consistency and integrity, and may also be used to facilitate an aspect of data repair relating to data-to-ontology annotation.

6. ACKNOWLEDGMENTS

We would like to acknowledge all members of the DIACHRON consortium for their contribution to this work. In particular we would like to thank Christos Pateritsas, Yannis Roussakis, Hasapis Panagiotis, Marios Meimaris and Giorgos Flouris for their contribution to components of the DIACHRON platform utilized by the OLS application.

7. REFERENCES

[1] Petryszak, R. et al. 2013. Expression Atlas update – a database of gene and transcript expression from microarray and sequencing-based functional genomics experiments. In *Nucleic Acids Research,* DOI=10.1093/nar/gkt1270.

[2] Welter D, MacArthur J, Morales J, Burdett T, Hall P, Junkins H, Klemm A, Flicek P, Manolio T, Hindorff L, and Parkinson H. 2014. The NHGRI GWAS Catalog, a curated resource of SNP-trait associations. In *Nucleic Acids Research,* Vol. 42 (Database issue): D1001-D1006.

[3] Auer S., et al. 2012. Diachronic linked data: towards long-term preservation of structured interrelated information". In Proc. of the First International Workshop on Open Data (WOD '12). ACM, New York, NY, USA, 31—39.

[4] Papavassiliou V., et al. 2009. On Detecting High-Level Changes in RDF/S KBs, Proceedings of the 8th International Semantic Web Conference, October 25-29

[5] Barry Smith et al. 2007. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. In *Nature Biotechnology* 25, 1251-1255. DOI=10.1038/nbt1346.

[6] Roussakis Y., Chrysakis I., Stefanidis K., Flouris G., Stavrakas Y. 2015. A Flexible Framework for Understanding the Dynamics of Evolving RDF Datasets. *International Semantic Web Conference* (1) 2015: 495-512