Crawl Me Maybe: Iterative Linked Dataset Preservation

Besnik Fetahu, Ujwal Gadiraju, and Stefan Dietze

L3S Research Center, Leibniz Universität Hannover, Germany
{fetahu, gadiraju,dietze}@L3S.de

Abstract. The abundance of Linked Data being published, updated, and interlinked calls for strategies to preserve datasets in a scalable way. In this paper, we propose a system that iteratively crawls and captures the evolution of linked datasets based on flexible crawl definitions. The captured deltas of datasets are decomposed into two conceptual sets: evolution of (i)*metadata* and (ii)the actual *data* covering schema and instance-level statements. The changes are represented as logs which determine three main operations: *insertions, updates* and *deletions*. Crawled data is stored in a relational database, for efficiency purposes, while exposing the *diffs* of a dataset and its live version in RDF format.

Keywords: Linked Data; Dataset; Crawling; Evolution; Analysis

1 Introduction

Over the last decade there has been a large drive towards publishing structured data on the Web. A prominent case being data published in accordance with Linked Data principles [1]. Next to the advantages concomitant with the distributed and linked nature of such datasets, challenges emerge with respect to managing the evolution of datasets through adequate preservation strategies. Due to the inherent nature of linkage in the LOD cloud, changes with respect to one part of the LOD graph, influence and propagate changes throughout the graph. Hence, capturing the evolution of entire datasets or specific subgraphs is a fundamental prerequisite, to reflect the temporal nature of data and links. However, given the scale of existing LOD, scalable and efficient means to compute and archive diffs of datasets are required.

A significant effort towards this problem has been presented by Käfer et al.[2], with the Dynamic Linked Data Observatory: a long-term experiment to monitor a two-hop neighbourhood of a core set of diverse linked data documents.

The authors investigate the lifespan of the core set of documents, measuring their on and off-line time, and the frequency of changes. Furthermore, they delve into how the evolution of links between dereferenceable documents over time. An understanding of how links evolve over time is essential for traversing linked data documents, in terms of reachability and discoverability. In contrast to the previous initiatives, in this work we provide an iterative linked dataset crawler.

It distinguishes between two main conceptual types of data: *metadata* and the actual *data* covering schema and instance-level statements.

In the remainder of this paper, we explain the schema used to capture the crawled data, the workflow of the iterative crawler and the logging states which encode the evolution of a dataset.

2 Iterative Linked Dataset Crawler

The dataset crawler extracts resources from linked datasets. The crawled data is stored in a relational database. The database schema (presented in Figure 1) was designed towards ease of storage and retrieval.

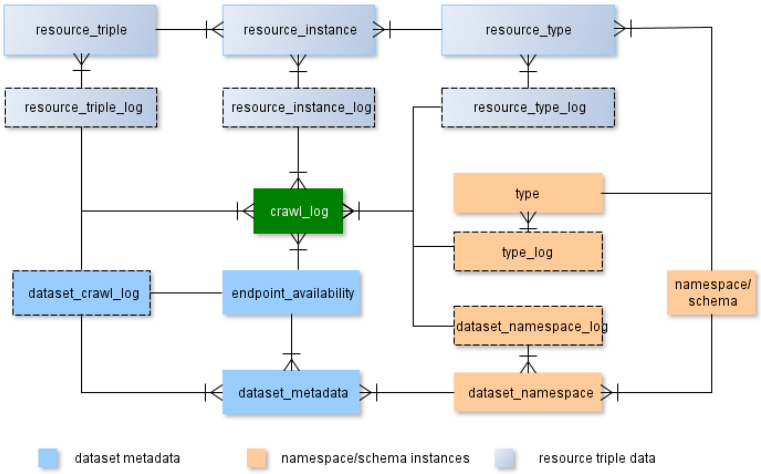

Fig. 1. Conceptual schema for the iteratively crawled linked datasets. Logs are represented with dashed lines (e.g. triple `insertion`: $\langle s, p, o \rangle$) of the various conceptual classes of data within linked datasets.

The crawler is designed with the intent to accommodate methods for assessing the temporal evolution of linked datasets. A dataset which has not been crawled before will thereby be crawled completely and all corresponding data will be stored in a database. This would thereby correspond to a dump of that dataset, stored according to the database schema. In case a dataset has already been crawled, the differences between the previously crawled state of the dataset and the current state are determined on-the-fly. Such Δs or *diffs*, are then stored. Therefore, for any dataset that has been crawled multiple times at different crawl-points[1], it is possible to reconstruct the state of the dataset at any of the given crawl-points.

2.1 Computation of Diffs

The differences between the state of a dataset at different crawl-points can be captured efficiently using the dataset crawler. Evolution of datasets can be

[1] The time at which a given *crawl* operation is triggered.

computed at different levels. Each crawl explicitly logs the various changes at schema and resource-levels in a dataset as either *inserted, updated* or *deleted*. The changes themselves are first captured at triple-level, and then attributed to either schema-level or resource instance-level. The following log operators with respect to dataset evolution are handled by the dataset crawler.

- **Insertions.** New triples may be added to a dataset. Such additions introduced in the dataset correspond to insertions.
- **Deletions.** Over time, triples may be deleted from a dataset due to various reasons ranging from persisting correctness to detection of errors. These correspond to deletions.
- **Updates.** Updates correspond to the update of one element of a triple $< s, p, >$.

Figure 2 presents an example depicting the computation of Δ between a previously crawled dataset at crawl-point t_0 and a fresh crawl at crawl-point t_1.

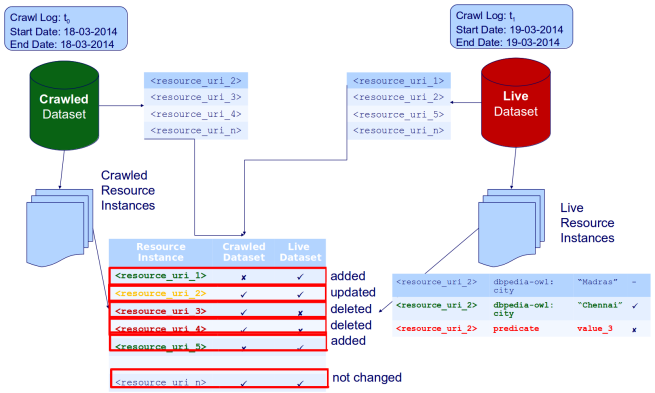

Fig. 2. Computation of *diffs* on-the-fly.

First, assume a change in the 'live dataset' in the form of an insertion of the triple corresponding to the URI `resource_uri_2`. Thus, the triple describing the city `Madras` is added. Consequently, if the value of the property `dbpedia-owl:city` is updated, then a subsequent crawl would capture this difference in the literal value of the property as an update to `Chennai`. Similarly, deletions made are also detected during the computation of *diffs*. Thus, computing and storing *diffs* on-the-fly in accordance with the log operators is beneficial; we avoid the overheads emerging from storing dumps of entire datasets.

2.2 Web Interface for the Iterative Dataset Crawler

We present a Web interface (accessible at `http://data-observatory.org/dataset_crawler`) that provides means to access the crawled resources, given specific crawl-points of interest from the periodical crawls. The interface allows us to filter for specific datasets and resource types. The Web application has three main components (see Figure 3): (i) displaying metadata of the dataset, (ii) dataset evolution, showing summaries of added/updated/deleted resources for

the different types, and (iii) dataset type-specific evolution, showing a summary of the added/updated/deleted resource instances for a specific resource type and corresponding to specific crawl time-points. In addition, the crawler tool is made available along with instructions for installation and configuration[2].

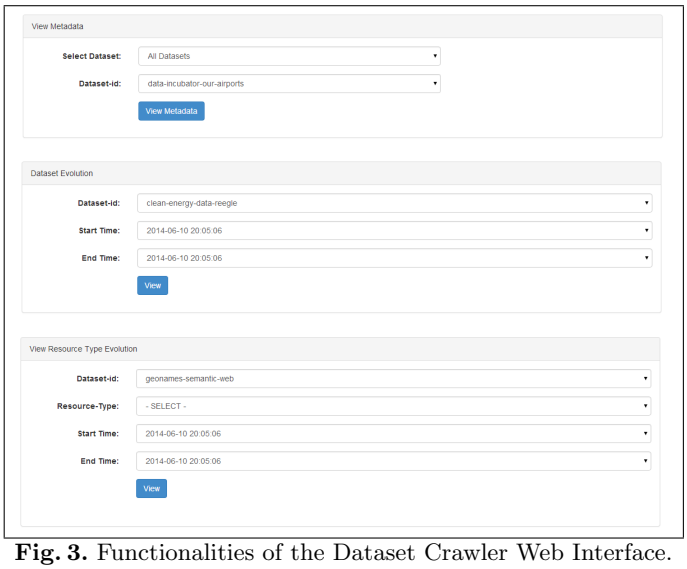

Fig. 3. Functionalities of the Dataset Crawler Web Interface.

3 Conclusion

In this paper, we presented a linked dataset crawler for capturing dataset evolution. Data is preserved in the form of three logging operators (insertions/updates/deletions) by performing an online Δ computation for any given dataset with respect to the live state of the dataset and its previously crawled state (if available). Furthermore, the crawled and computed Δ of a dataset can be used to assess its state at any given crawl-point. Finally, we provided a web interface which allows the setup of the crawler, and facilitates simple query functionalities over the crawled data.

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
2. T. Käfer, A. Abdelrahman, J. Umbrich, P. OByrne, and A. Hogan. Observing linked data dynamics. In *The Semantic Web: Semantics and Big Data*, pages 213–227. Springer, 2013.

[2] `https://github.com/bfetahu/dataset_crawler`