Observing Linked Data Dynamics

Tobias Käfer[1], Ahmed Abdelrahman[2], Jürgen Umbrich[2],
Patrick O'Byrne[2], and Aidan Hogan[2]

[1] Institute AIFB, Karlsruhe Institute of Technology, Germany
[2] Digital Enterprise Research Institute, National University of Ireland Galway

Abstract. In this paper, we present the design and first results of the *Dynamic Linked Data Observatory*: a long-term experiment to monitor the two-hop neighbourhood of a core set of eighty thousand diverse Linked Data documents on a weekly basis. We present the methodology used for sampling the URIs to monitor, retrieving the documents, and further crawling part of the two-hop neighbourhood. Having now run this experiment for six months, we analyse the dynamics of the monitored documents over the data collected thus far. We look at the estimated lifespan of the core documents, how often they go on-line or off-line, how often they change; we further investigate domain-level trends. Next we look at changes within the RDF content of the core documents across the weekly snapshots, examining the elements (i.e., triples, subjects, predicates, objects, classes) that are most frequently added or removed. Thereafter, we look at how the links between dereferenceable documents evolves over time in the two-hop neighbourhood.

1 Introduction

The Web of (Linked) Data is unquestionably dynamic: over time, documents come online, documents go offline, and the content of online documents changes. However, the dynamics of Linked Data are not yet well understood in terms of how stable documents are over time, what kinds of changes are most frequently encountered, and so forth. Knowledge about Linked Data dynamics is important for a wide range of applications: effective caching, link maintenance, versioning, etc. The current lack of understanding about Linked Data dynamics can be attributed to a lack of suitable collections to analyse: to track changes over time, and to ultimately derive meaningful results about the dynamics of Linked Data, we need to monitor a fixed set of diverse Linked Data documents at a fixed interval over a long period of time. As of yet, no such collection has been made available to the Linked Data research community.

In this paper, we aim to shed light on the dynamics of Linked Data. We first present some use-cases to motivate why knowledge about the dynamics of Linked Data is important to the community (§ 2). We then introduce the Dynamic Linked Data Observatory (DyLDO), which we have created to monitor a fixed set of Linked Data documents (and their neighbourhood) on a weekly basis for an indefinite period of time: we discuss our methodology for selecting

documents and for monitoring these sources (§ 3). After six months of monitoring, we analyse the 29 weekly snapshots collected and analyse the dynamics exhibited by documents in the collection. We look at: (§ 4) the stability and lifespan of documents in the snapshots, and how often their content changes; and (§ 5) the types of changes these documents undergo: are they additions or deletions, what elements of the RDF document changed, and so forth.

This paper is a continuation of a previous workshop paper [5], where we originally motivated and outlined the methodology for our Dynamic Linked Data Observatory. Herein, we summarise discussion on the observatory and focus on our first concrete results for Linked Data dynamics after 29 weeks of monitoring.

2 Motivation and Novelty

We first discuss a brief selection of use-cases to help motivate our work on Linked Data dynamics and its importance to the community.

Focused synchronisation: Various centralised search & query approaches for Linked Data rely on locally replicated copies of RDF harvested from the Web. As the original sources change, replicated indexes become stale, affecting the up-to-dateness of results. More fine-grained knowledge about Linked Data dynamics would allow centralised engines to, e.g., focus on keeping synchronised with those domains whose contributions change rapidly.

Smart caching: Conversely, "live querying" approaches for Linked Data dereference and discover sources on the fly. However, remote lookups are expensive to execute. Knowledge about Linked Data dynamics can help to identify which sources can be cached to save time and resources, how long cached data can be expected to remain valid, and whether there are dependencies in the cache (e.g., if a document from a particular domain changes, should all documents from that domain be invalidated?).

Hybrid architectures: A core engineering trade-off for systems dealing with lots of data is pre-processing overhead vs. runtime-processing overhead. In general, pre-processing (e.g., caching, local indexing, reasoning materialisation, etc.) is better suited to static data, whereas runtime-processing (e.g., live dereferencing, backward-chaining, etc.) is better suited to dynamic data. In a hybrid architecture, knowledge about dynamics can be used to delegate both data and requests into static/dynamic pipelines. Static data can be cached and deeply pre-processed, whereas dynamic requests may invoke a "live querying" component or backward-chaining reasoning, and so forth.

Link maintenance: When Linked Data publishers embed links to external domains in their data, deadlinks will occur after some time, or links may no longer be appropriate after remote data changes. Furthermore, novel sources may serve as useful targets to link. Knowledge about dynamics can help publishers to decide how frequently their link-sets need to be updated depending on, e.g., the domain they target or the type of link.

Versioning: When changes are made to a dataset, versioning should be applied to ensure that parties relying on the data in question do not suffer

adverse effects (e.g., through use of deprecation instead of simply removing data). Versioning is particularly relevant for vocabularies on the Web, whose semantics may change over time to reflect usage. Knowledge about Linked Data dynamics can show how changes propagate on the Web and inform the design of mature versioning methodologies.

In terms of existing works, various papers on the dynamics of the HTML-centric Web have been published by, e.g., Coffman et al. [3], Brewington and Cybenko [1], Lim et al. [8], Cho and Garcia-Molina [2], Fetterly et al. [4] and Ntoulas et al. [9]. These works analysed the rate of change of documents, patterns in change (e.g., time of day, day of the week), growth rate of the Web, dynamicity of links, the relation between top-level domains and dynamicity, etc. We refer readers to the broad survey by Ke et al. [6] about Web dynamics. As opposed to these related works, we focus specifically on the dynamicity of RDF documents in the context of Linked Data.

Few papers specifically analyse RDF or Linked Data dynamics. Popitsch and Haslhofer [10] propose DSNotify to help maintain links between datasets, but only have knowledge of DBpedia dynamics. In previous work, we showed that two centralised query indexes of Linked Data (OpenLink's LOD Cache[3] and Sindice's SPARQL endpoint[4]) often return stale results [13]. In another previous work, we analysed changes in documents over 24 snapshots of RDF Web data [12]; however, the coverage of each snapshot varied and our analysis was rather "best-effort". Addressing this problem, we later proposed the Dynamic Linked Data Observatory [5] to collect the snapshots upon which this work is based.

3 Dynamic Linked Data Observatory

To study the dynamics of Linked Data in a principled way, we require a principled way of monitoring a sample of Linked Data documents over time. Given a lack of suitable data available elsewhere, earlier this year we proposed and implemented the Dynamic Linked Data Observatory to perform this monitoring [5]. Each week, a fixed set of documents is retrieved and the content stored. From this core set of documents, we perform a brief crawl to find well-linked documents in their close neighbourhood. We began the weekly monitoring experiments on 2012/05/06, and have collected 29 snapshots until the time of writing. Herein, we outline our methodology for sampling and monitoring documents. *Full details of our sampling and crawling configurations are available in [5].*

3.1 Sampling Methodology

We wish to monitor a broad cross-section of Linked Data documents for a sample that would lead to manageable weekly snapshot sizes and that would not overburden publishers with repetitive deep crawls. In February 2012, we extracted

[3] http://lod.openlinksw.com/sparql; retr. 2013/03/12.
[4] http://sparql.sindice.com/; retr. 2013/03/12.

the list of 220 URIs available on the DataHub site under the "LOD cloud" group, offering entry points for (most) of the datasets listed in the LOD cloud.[5] To this, we added the top-220 documents extracted from the Billion Triple Challenge (BTC) 2011 dataset as determined by PageRank over the graph of dereferenceable documents. These initial 440 URIs offer core entry points into both the LOD cloud and BTC perspectives of the Web of Data (see [5] for details).

From these 440 URIs, we then wished to expand our sample by means of a crawl that would stay in the vicinity of our core URIs (and, e.g., avoid getting trapped in high-volume exporters with low out-degrees such as the hi5.com or livejournal.com sites). We thus performed a 2-hop breadth first crawl using the 440 URIs as our seed-list, considering all URIs mentioned in an RDF document as a potential link, looking for RDF/XML, RDFa, N-Triples or Turtle content, enforcing a two-second politeness delay between lookups to the same site. We repeated this crawl 10 times to account for the possibility of non-determinism and instability of hosted documents. We then took the union of all URIs that dereferenced to RDF content in one of the crawls, resulting in a core monitoring set of 95,737 dereferenceable URIs spanning 652 pay-level domains[6], giving an average of 146.8 dereferenceable URIs per domain (see [5] for full details).

3.2 Monitoring Methodology

The core aim of the weekly monitoring setup is to dereference and download the content for the list of 95,737 URIs sampled in the previous step. Since this set of documents is static, we also extend this "kernel" of monitored data by crawling a further 95,737 URIs starting from the kernel. The content of this extended crawl varies from week to week, and captures new documents in the neighbourhood of the kernel, as well as raw data reflecting changes in the link-structure of the kernel. The extension of the kernel is done by breadth-first crawl, and involves at least 2 hops (sometimes 3 hops) to meet the quota.

We have performed this monitoring on a weekly basis since 2012/05/06, yielding 29 weekly snapshots at the time of writing. Each snapshot consists of the content retrieved for the core kernel URIs (following redirects), the content of the expanded crawl, a set of redirects, and access logs for URIs that were looked up. Table 1 enumerates the average and total amount of data retrieved over the 29 weeks for the kernel documents and for the expanded crawl. The 95,737 kernel URIs yielded an average of 68,997 documents: though all URIs were deemed to dereference to RDF during sampling, some dereference to the same RDF document and some now fail to dereference. The number of unique documents appearing in at least one kernel snapshot was 86,696 and the analogous figure for domains was 620 (vs. 652 for the source URIs). In terms of the diversity of the kernel, the documents in each snapshot came from an average of 573.6 domains. The sum of all kernel snapshots yields around 464 million quadruples. By

[5] http://thedatahub.org/group/lodcloud; retr. 2013/03/12.

[6] The level of domain which an agent can register and must pay for: e.g., dbpedia.org, bbc.co.uk. We may refer to pay-level-domains as PLDs or as simply "domains".

Table 1. Overall statistics across all 29 snapshots

Statistic	Kernel	Extended
Mean pay-level domains	573.6 ±16.6	1,738.6 ±218
Mean documents	68,996.9 ±5,555.2	152,355.7 ±2,356.3
Mean quadruples	16,001,671 ±988,820	94,725,595 ±10,279,806
Sum quadruples	464,048,460	2,747,042,282

comparison, the extended snapshots contain 3× the number of domains, 2.2× the number of documents, and 5.9× the amount of raw data (there was thus a higher statement per document ratio in the extended crawl). In this paper, we currently focus on a first analysis of changes within the kernel documents.

4 Document-Level Dynamics

In this section, for the documents retrieved from the fixed set of kernel URIs, we first look at the availability of documents over time, the estimated life-span and death-rate of these documents, and their rates of change.

4.1 Availability/Occurrence

As aforementioned, 86,696 RDF documents appeared in (i.e., returned content for) at least one kernel. Figure 1 shows the distribution of the availability of these documents, counting for how many snapshots they appeared, measuring their stability over the 29 weeks. We see that 26% were available for all 29 weeks of the monitoring period. 55% of documents were available for 27 weeks or more and the mean availability for documents was 23.1 snapshots (79.7% availability).

With respect to this "one-in-five" unavailability of documents, Figure 2 provides a breakdown of the HTTP response codes and errors encountered while accessing URIs in the kernel (after following redirects). Response codes in `2xx` are most common: all of these were `200 Okay` indicating content was returned. The remaining responses indicate errors, where we see increasing instability over the monitoring time-frame. Most errors were `5xx` server error codes, the most common (32%) of which were `500 Internal Server Error`. The "other" category of errors related to unknown hosts and other HTTP-level inconsistencies such as self-redirects. A small but growing number of errors were `4xx` codes, 96% of which were specifically `404 Not Found`, indicating that there is no longer any document at that location. We next investigate these "dead documents".

Discussion: A one-in-five unavailability rate suggests that an agent traversing Linked Data documents can, on a single pass, expect to miss about 20% of potential content. This unavailability is not unique to Linked Data: for example, looking at 151 million HTML pages in 2003, Fetterly et al. [4] managed to download only 49.2% of pages eleven times in eleven weeks; in fact, our results are much more stable by comparison (cf. [4, Figure 4] and Figure 2). One may then ask how often unavailability is temporary, rather than permanent.

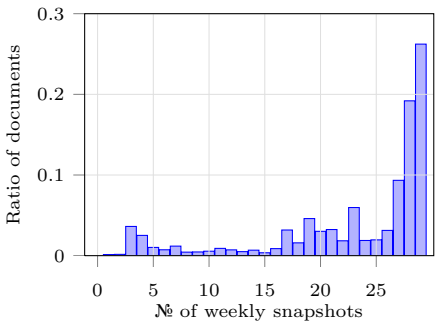

Fig. 1. Appearances of documents

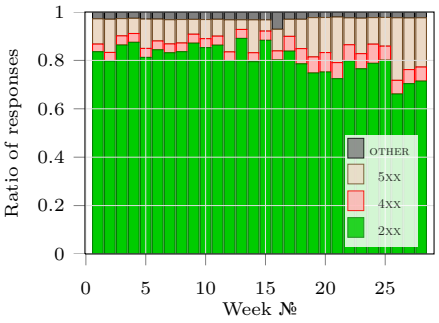

Fig. 2. Response distributions

4.2 Death Rate

Given estimates about their stability, we now estimate the loss of documents in the kernel over time by identifying *dead documents*: documents that (are likely to) have gone permanently offline. First, we look at the *last-heartbeat* of documents: the last weekly snapshot in which the document appeared such that, e.g., if a document was last seen in week 2, this document is unlikely to ever return. Figure 3 shows the evolving last heart-beats of kernel documents where, e.g., we see that 95% of documents have appeared at least once since the 14^{th} snapshot (2012/08/05). The further left the life-span, the longer the document is offline and the less likely that it will return. Thus the sharp downward trend observable for the last three snapshots could be due to temporary issues.

Taking another perspective, we also estimate the death-rate of documents by looking specifically at `404` errors that often indicate a permanent error (vs. `5xx` codes that may indicate, e.g., temporary unavailability or bugs). We found that 98.3% of URIs that return a `404` error never return content again in our monitoring frame, and 99.7% of URIs that return two sequential `404` errors never return. Based on returning a sequence of `404` codes up to the most recent snapshot, Figure 4 shows the rate at which documents die in a manner comparable with the analogous "last heart-beat" measures: the `404` death-rate likely underestimates the amount of dead documents (since it does not cover all possible causes), whereas the last heart-beat measure certainly overestimates the amount of dead documents. Combining both perspectives, 5% of documents have returned a trailing sequence of five or more `404`s or have been offline for more than 14 weeks, strongly indicating death.

Discussion: The one-in-twenty death-rate of Linked Data documents over six-months is pertinent for link-maintenance (detecting and avoiding dead-links) and for cache maintenance. The death-rate of 5% over six months can be compared favourably with death-rates of 20.5% observed by Koehler [7] in 1999 and 48% observed by Ntoulas et al. [9] in 2004 for HTML documents. We conjecture that since (cool) URIs also serve as names in Linked Data, such documents often have more stable URLs than, e.g., HTML URLs that often contain query strings.

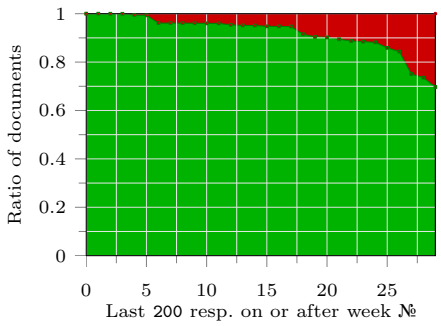

Fig. 3. Last heartbeat of documents

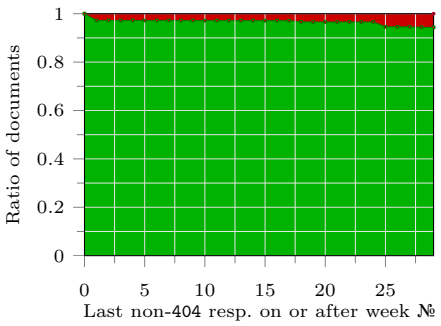

Fig. 4. Documents reported dead

4.3 Change Ratio

Next we compare the RDF content of the documents on a week-to-week basis. For each document, we compare 28 sequential version pairs. If a document was not available for a given week, we make the comparison with the most recent available version of the document. We wish to compare RDF content and not document syntax: thus, our comparison is complicated by the presence of existential blank nodes. In theory, our crawler uses a deterministic mechanism for labelling blank-nodes such that, for a given Web document, the labels of blank nodes will be consistent if blank nodes are consistently labelled in the source document and/or the order of implicit blank nodes remains the same. However, some documents in our collection provide fresh, explicit blank node labels upon every access.[7] Henceforth, when comparing graphs, we apply an approximation whereby we rewrite all blank nodes to a single, global, fresh constant (i.e., we considering all blank nodes as equal). This allows us to detect changes in documents, including additions and deletions of statements, irrespective of blank node labels. We compared this approximation to an isomorphism check for RDF graph equivalence and found that it corresponded in all pair-wise comparisons of document versions for both positive and negative cases.

The distribution of changes for the kernel documents across the 29 snapshots is plotted in Figure 5, where we see the ratio of documents with 0–28 changes across the snapshot pairs. At $x = 0$, we see that 62.2% of documents did not change over the 29 weeks. Thereafter, we see that most other documents changed infrequently or changed very frequently: 23.2% fell into the slightly dynamic $[1, 3]$ interval, 8.4% fell into the highly dynamic $[24, 28]$ interval, and 6.2% fell into the large remaining $[4, 23]$ middle interval.

Next, we are interested to characterise changes of documents within the same pay-level-domain. In Figure 6, we plot domains along two dimensions of change: the x-axis represents the ratio of documents on that domain that exhibited at least one change in the monitoring period, the y-axis represents the mean

[7] See, e.g., `http://dbtune.org/artists/last-fm/Baracudas.rdf`; retr. 2013/02/12.

number of changes for the documents on that domain (including only those that changed at least once), and the size of the tick indicates the number of sampled documents for the domain. We also annotate some examples of notable domains. Many domains sit at the origin indicating no changes in any document. Relatedly, since the majority of domains tend to cluster towards three of the four corners, we can consider the following classification of domains:

STATIC domains contain a low ratio of documents that change, and these documents change infrequently. Per Figure 6, 322 domains fell into the STATIC quadrant (**51.9%**), including linkedmdb.org, bbc.co.uk, w3.org, etc.

BULK domains contain a high ratio of documents that change, but these documents change infrequently. Per Figure 6, 182 domains fell into the BULK quadrant (**29.4%**), including dbpedia.org, freebase.com, bio2rdf.org, etc.

DUAL domains contain a low ratio of documents that change, but these documents change frequently. Per Figure 6, only 6 domains fell into the DUAL quadrant (**1.0%**), including loc.gov and geospecies.org.

ACTIVE domains contains a high ratio of documents that change, and these documents change frequently. Per Figure 6, 110 domains fell into the ACTIVE quadrant (**17.7%**), including dbtropes.org, dbtune.org, linkeddata.es, etc.

We highlight that for many of the BULK domains, although a large number of documents changed in the course of our observations, all changes for these domain tended to happen together: for such domains, the median number of weeks with changes was 4 (with no change on the domain between 24 weeks).

Based on meta-data from the LOD cloud and the DataHub[8], in Table 2, we show the breakdown of domains in the categories outlined above for (i) dataset topic, and (ii) whether the data is exported directly by the producer or by a third party. We could not locate topic or producer information for many (non-LOD) domains with few documents (cf. Table 2). Since domains may host multiple datasets, if we found multiple topics or production types associated to a single domain, we categorised it as cross-domain or *both*, respectively. In general, we see few high-level patterns in dynamicity for different topics or methods of production. Perhaps most notably, third-party exporters tend to be more active than first-party producers (presumably due to "live exporters"). Also, user-generated domains tended to be more active (though the number of such domains was low).

Discussion: We find that 62.2% of documents did not change in the 29 weeks and thus are obvious candidates for long-term caching. This compares with, e.g., 56% of static HTML pages reported by Brewington and Cybenko [1] in 2000, 65.5% reported by Fetterly et al. [4] in 2003 and 50% reported by Ntoulas et al. [9] in 2004. Such works also confirm that past dynamicity can be used to predict future dynamicity. Our work also clusters changes per domain, helping to design synchronisation strategies, where, e.g., a change detected for a BULK site such as dbpedia.org suggests that all documents from that domain should be refreshed. Similarly, Ntoulas et al. [9] showed that change predictions made for individual sites can often (but not always) be accurate.

[8] http://lod-cloud.net; http://datahub.io/; retr. 2013/03/08.

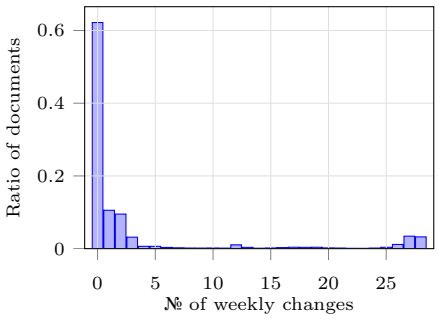

Fig. 5. Document change distribution

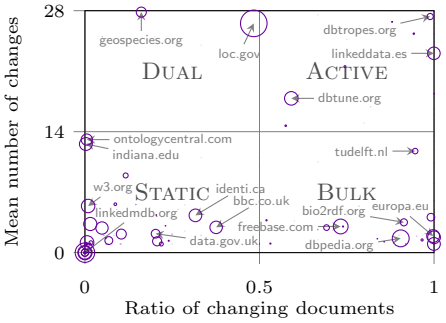

Fig. 6. Clustering of domain changes

Table 2. Dynamicity of Linked Data domains per topic and per party involved

Category	Doc №	Dom №	Static		Bulk		Dual		Active	
			№	%	№	%	№	%	№	%
cross-domain	34,872	33	21	63.64	6	18.18	2	6.06	4	12.12
geographic	4,693	10	6	60.00	2	20.00	1	10.00	1	10.00
government	5,544	14	10	71.43	3	21.43	0	0.00	1	7.14
life-sciences	2,930	4	2	50.00	2	50.00	0	0.00	0	0.00
media	8,104	10	6	60.00	2	20.00	0	0.00	2	20.00
publications	14,666	35	24	68.57	8	22.86	2	5.71	1	2.86
user-generated	7,740	12	7	58.33	0	0.00	0	0.00	5	41.67
unknown	8,147	502	246	49.00	159	31.67	1	0.20	96	19.12
first-party	22,649	50	38	76.00	8	16.00	2	4.00	2	4.00
third-party	29,078	61	37	60.66	12	19.67	1	1.64	11	18.03
both	27,520	23	13	56.52	6	26.09	2	8.70	2	8.70
unknown	7,449	486	234	48.15	156	32.10	1	0.21	95	19.55
total	86,696	620	322	51.94	182	29.35	6	0.97	110	17.74

5 RDF-level Dynamics

We see that Linked Data documents change with varying degrees of breadth and frequency on different domains, and that documents on some domains, such as dbtropes.org, change widely and often. We now look at what kinds of changes are occurring on an RDF-level within these documents.

5.1 Types of Triple-Level Changes

We first look at the types of changes for documents. We found that 27.6% of documents only ever updated values for terms (one per triple) in the RDF graph they contain across the 29 weeks, keeping the number of triples static: such changes would include, e.g., updating a literal value like as an access-date entry. A further 24.0% of documents only added triples across the 29 weeks, representing monotonic additions. Changes for other documents involved a mix of additions, single-term updates and deletions across the different weeks.

In Figure 7, we plot the ratio of documents for which we found at least one triple addition over the 29 weeks against the ratio of documents for which we

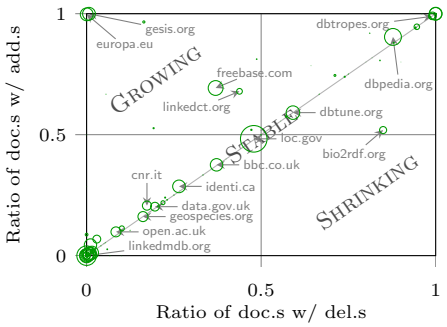

Fig. 7. Ratio of documents with additions vs. deletions per domain

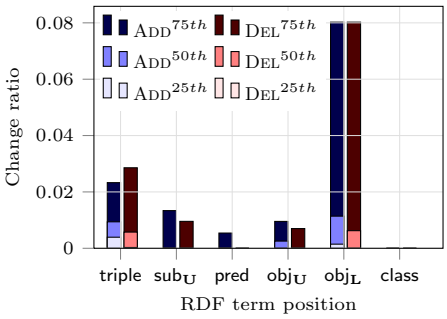

Fig. 8. Additions (left) and deletions (right) for different RDF elements

encountered some deletion over the 29 weeks, looking for high-level patterns. For the purposes of this plot, we consider a term update as an addition and a deletion of a triple. We see that most of the domains fall along a stable line where an equal number of documents involve some additions and some deletions: again, many of these documents correspond to the 27.6% that only update individual values (an add and a delete). Close to the $(0, 1)$ point, we see two large "monotonic" domains (europa.eu and gesis.org) that almost always only ever *add* triples to their documents. The one notable domain in the SHRINKING area was bio2rdf.org, for which 52% of documents had additions and 85% had deletions.

Discussion: For Linked Data warehouses, additions are often cheaper than deletions (esp. if, e.g., inference and truth maintenance are required). Here we see that additions to Linked Data documents are almost always accompanied by deletions, emphasising the importance of efficient revision strategies for warehouses. In relation to the HTML Web, Brewington and Cybenko [1] show that the content of HTML pages tends to grow over time, though their results rather reflect technological trends over a period of many years ('95–'99).

5.2 Types of Term-Level Changes

Next we look at the types of terms changing in the RDF content of the kernel. Figure 8 plots the 25^{th}, 50^{th} and 75^{th} percentiles[9] for the addition/deletion of RDF triples and the terms they contain. We only consider documents that changed at least once in the 29 weeks and omit blank node terms due to possible homomorphisms (relying on our approximation for *triples* involving blank nodes). We compare changes to subject URIs, predicates, object URIs, object literals and classes (values for `rdf:type`). The y-axis reflects the ratio of triples or terms that changed versus the total number of unique such elements observed in the documents considered (the y-range is small: $[0, 0.08]$). A ratio of 0.08 for object literal additions thus indicates that, over 29 weeks, the number of unique

[9] Higher percentiles cause too much compression of the results; hence we omit them.

object literals added to the documents at that percentile was 0.08× the total number of unique object literals appearing in those documents.

We see some clear trends. First, we see that additions and deletions are often of a similar magnitude, reflecting back on previous observations about terms often being directly replaced. Second, the most dynamic position of an RDF triple is the object, with a high churn of object literal values in particular. Conversely, predicates are occasionally added but rarely removed from documents. Analogously, class terms are very rarely added and very rarely removed (barely even seen above the x-axis). These latter two observations suggest that the *schema signature* of documents (set of property/class terms used) is generally static.

Discussion: The types of terms that change offer interesting high-level patterns into the dynamicity of RDF in general. For example, the observation that the set of properties and classes instantiated by a document rarely changes lends empirical strength to proposals for schema-level summaries of data, such as proposed by Stuckenschmidt et al. [11]. On the other hand, we see that literals are the most dynamic element of RDF. The following section sheds light on why this might be the case.

5.3 Dynamic Predicates

Though we have seen that predicates themselves are rarely added or removed, we are interested to see which predicates are indicative of dynamic statements. Table 3 presents the ten most dynamic predicates according to the ratio of added (+) and deleted (−) statements involving that predicate, divided by the total number of statements for that predicate across all snapshots; we only include predicates that appear in all snapshots and appear in $\geq 1,000$ statements overall.[10] Where added and deleted ratios correspond closely, this suggests frequent "value updates". The two `dbtont:` predicates are used on the third-party dptropes.com domain to indicate a time-stamp since the relevant data were parsed or fetched from the original source (tvtropes.org); the `swivt:`, `prv:` and `linkedct:` predicates also provide time-stamps indicating the last time data were refreshed for documents on various domains. The two `sioc:` predicates are used to track dynamic discussions and posts on the social gnoss.com domain. The `media:image` predicate appears for RDFa image meta-data, most of which are embedded in msn.com news pages. The `xhtml:bookmark` predicate represents links embedded as RDFa in various dynamic XHTML pages.

Discussion: Identifying dynamic predicates allows warehouses to know, in a granular fashion, which parts of an input query relate to static knowledge and which parts to dynamic knowledge (e.g., see our previous proposals on this topic [13]). Per our results, when considering cached content, the ratio of additions indicates the potential to miss answers involving triples with a given predicate, and the ratio of deletions indicates the potential to return stale answers. With respect to the most dynamic predicates, we identify that they are

[10] Prefixes can be found at `http://prefix.cc`; retr. 2013/03/12.

Table 3. Top-10 dynamic predicates ('*' indicates `provenance_time_updated` and `provenance_time_added`, respectively)

№	Predicate	Total	+	−
1	`dbtont:parsed`	35,911	0.94	0.94
2	`sioc:has_discussion`	3,171	0.87	0.99
3	`sioc:content`	107,387	0.87	0.98
4	`dbtont:fetched`	34,894	0.53	0.53
5	`swivt:creationDate`	35,295	0.53	0.53
6	`media:image`	1,377	0.49	0.49
7	`prv:performedAt`	16,706	0.45	0.45
8	`xhtml:bookmark`	17,852	0.45	0.44
9	`linkedct:p.t.u`*	2,652	0.42	0.42
10	`linkedct:p.t.a`*	2,652	0.42	0.42

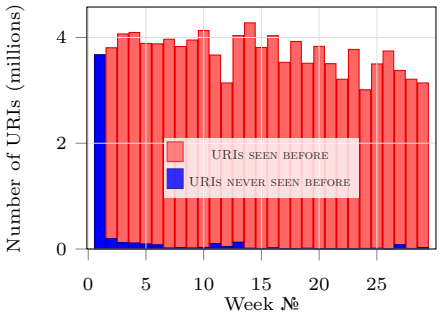

Fig. 9. Links extracted from kernels

often trivial time-stamps. Comparatively, Fetterly et al. [4] and Ntoulas et al. [9] both discuss how the majority of changes in HTML documents are very minor, involving hit counters, time-stamps, etc.

5.4 RDF Link Structure

Finally, we look at the evolving nature of the link structure of documents over time. We first want to see if the overall level of links tends to increase or decrease over time, and are interested to see at what rate fresh links are added to the kernel. We consider any URI in any position of a triple as a potential link from the kernel. Figure 9 plots the evolution of the volume of such links over time. We see that the number of links can fluctuate based on the availability of documents (with parallels to, e.g., response code distributions for each week illustrated in Figure 2). A second key observation is that the ratio of fresh URI links added to the kernel is in general quite small: we consider a URI as fresh if it has not been seen for *any* kernel snapshot before. This clearly indicates that the outward link structure of the kernel remains quite static (aside from instability) over time. In fact, if anything, links are on a decreasing trend as documents die off.

That said, after the initial stabilisation over the first month of observations, we do find that a few domains are consistently contributing some fresh links to the kernel: sec.gov, identi.ca, zitgist.com, dbtropes.org, ontologycentral.com and freebase.com offer a low-volume but constant stream of fresh outward links from week to week. Other domains—including bbc.co.uk, bnf.fr, dbpedia.org, linkedct.org, bio2rdf.org, etc.—tend to introduce new links in batches, corresponding with the update characteristics of domains plotted previously in Figure 6. However, such domains are the exception rather than the rule.

Discussion: Knowledge about how links change over time is important for any agent that traverses Linked Data documents (in terms of reachability, discoverability, etc.) or analyses link structure (e.g., to compute PageRank), etc. Ntoulas et al. [9] found that hyperlinks in HTML documents tend to be more dynamic than other forms of content, estimating that 25% of links are new each week (though considering a growing set of documents). In comparison, our results

seem much more static. This seems counter-intuitive in that Linked Data itself is fundamentally comprised of URIs and thus links; however, we already saw that URI terms in RDF documents change slowly (compared to, e.g., literals).

6 Conclusions

Six years on from the original publication of the Linked Data principles, we present some first concrete results on the dynamics of Linked Data documents.

Our first contribution is the design, implementation and upkeep of the Dynamic Linked Data Observatory. We have been running this observatory since May 2012 and have collected a significant corpus of data that captures the inherent dynamics of Linked Data. We will continue to run this experiment indefinitely, allowing us to draw further conclusions over the data. We make all data available for the community; please see `http://swse.deri.org/dyldo/` for up-to-date weekly snapshots. In the near future, we plan to extend this site with live statistics, meta-data and APIs.

Our second core contribution is the comprehensive analysis of the dynamics of Linked Data presented here. Based on monitoring 86,696 Linked Data documents for 29 weeks, we found that documents were unavailable 20% of the time and further estimated that 5% of documents had gone permanently offline in that period. We then determined that 62.2% of documents had no change in that time, where other documents either changed very frequently (8.4%), or very infrequently (23.2%), with few documents in between. Of the documents that did change, many updated individual RDF terms in the document (27.4%) or only ever added triples (23.1%). We found that domains tended to be either very static (44.5%), have a high ratio of documents that change infrequently (28.2%), or have a high ratio of documents that change frequently (25%); most domains contain a balance of documents with additions and deletions. With respect to the types of changes occurring on an RDF level, we found that object literals were the most liable to change ($0.01\times$ ratio for median/50^{th} percentile; $0.08\times$ for 75^{th} percentile), whereas the schema signature of documents—involving predicates and values for `rdf:type`—changed very infrequently. We identified predicates involved in the highest ratio of new/removed triples and found that they often relate to time-stamps. Finally, we showed that the rate of fresh links being added to the documents is low, varying between 4,960–126,944 depending on bulk domain updates that week.

In terms of connecting these observations back to our original use-cases outlined in Section 2, we make the following observations:

Focused synchronisation: We identified the general rate of change of documents, and found that dynamicity tended to follow certain predictable patterns for PLDs. For example, static domains infrequently require light synchronisation, bulk domains occasionally require heavy synchronisation, dual domains require frequent light synchronisation, active domains require frequent heavy synchronisation (or live querying techniques), etc.

Smart caching: Reversing the previous use-case, we found that 62.2% of documents didn't change over the six months and found that 51.9% of domains were considered static (and thus are candidates for long-term caching). Applications that rely on a schema-level index or schema-level cache of documents can rest assured that the schema-signature of documents tends to be very (though not completely) static. Furthermore, we identified particular predicates whose triples should not be cached due to high rates of updates.

Hybrid architectures: A hybrid architecture could be built along a number of logical data partitions. First, we showed that domains tend to fall into a few clusters, where static and bulk domains could be supported by heavy materialisation approaches, whereas active domains are best supported through decentralised live-querying approaches. Conversely, we also showed, for example, that different schema patterns in the data were indicators of different levels of dynamicity, where partitioning could be done on a per-predicate basis instead, etc.

Link maintenance: We found instability in documents, though much of this instability was of a temporary nature. However, we found that 5% of documents had died off during our monitoring period, suggesting an initial estimate for the arrival of deadlinks.

Versioning: We have not tackled the issue of versioning in depth. Some conclusions could be applied incidentally to the area of versioning (e.g., about the frequency of change of different types of RDF terms and the balancing of additions vs. deletions), but further more specialised analyses of the data (by us or the community) would be needed to generate concrete guidelines.

As highlighted by the last use-case, there is still further work to do.

7 Future Directions

In this paper, we focused on analysis of the kernel documents since they were retrieved through a consistent (and thus comparable) set of URIs. In future work, we would like to leverage the extended datasets, in particular to look at how often new RDF documents arise in the neighbourhood of the kernel.

A shortcoming of our current work is that we cannot say anything about changes at levels more fine-grained than a week. In our original proposal for the Dynamic Linked Data Observatory, we proposed to implement dynamic monitoring of documents that changed each week in increasingly more fine-grained intervals. We have yet to implement this feature, though this would give us knowledge of intra-week dynamics for (at least) a small number of highly-dynamic sources.

Finally, at some stage, we may need to consider an incremental extension of our kernel to include new Linked Data sources that are coming online. Our idea at the moment would involve infrequently adding 20% of fresh URIs on top of the kernel, possibly on a yearly basis. In general, we are open to extending our monitoring while maintaining the core kernel snapshots.

Acknowledgements. This paper was funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). We thank our anonymous reviewers (both LDOW and ESWC) for their helpful comments. We also thank Andrew Gallagher and Gerard Conneely at DERI, and the OpenCirrus team at KIT's Steinbuch Centre for Computing, for technical and infrastructural help.

References

1. B. Brewington and G. Cybenko. Keeping up with the changing web. *Computer*, 33(5):52–58, 2000.
2. J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3):256–290, Aug. 2003.
3. E. G. Coffman, Jr., Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of scheduling*, 1:0–21, 1997.
4. D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of Web pages. In *WWW*, pages 669–678. ACM, 2003.
5. T. Käfer, J. Umbrich, A. Hogan, and A. Polleres. DyLDO: Towards a Dynamic Linked Data Observatory. In *LDOW at WWW*. CEUR-WS Vol. 937, 2012.
6. Y. Ke, L. Deng, W. Ng, and D. L. Lee. Web dynamics and their ramifications for the development of Web search engines. *Computer Networks*, 50(10):1430–1447, 2006.
7. W. Koehler. An analysis of Web page and web site constancy and permanence. *Journal of the American Society for Information Science*, 50(2):162–180, 1999.
8. L. Lim, M. Wang, S. Padmanabhan, J. Vitter, and R. Agarwal. Characterizing Web document change. In *WAIM*, pages 133–144. Springer, 2001.
9. A. Ntoulas, J. Cho, and C. Olston. What's new on the Web? The evolution of the Web from a search engine perspective. In *WWW*, pages 1–12. ACM, 2004.
10. N. Popitsch and B. Haslhofer. DSNotify – a solution for event detection and link maintenance in dynamic datasets. *J. Web Sem.*, 9(3):266–283, 2011.
11. H. Stuckenschmidt, R. Vdovjak, G.-J. Houben, and J. Broekstra. Index structures and algorithms for querying distributed RDF repositories. In *WWW*, pages 631–639. ACM, 2004.
12. J. Umbrich, M. Hausenblas, A. Hogan, A. Polleres, and S. Decker. Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources. In *Proc. of LDOW at WWW*. CEUR-WS Vol. 628, 2010.
13. J. Umbrich, M. Karnstedt, A. Hogan, and J. X. Parreira. Hybrid SPARQL Queries: Fresh vs. Fast Results. In *ISWC*, pages 608–624. Springer, 2012.