Towards Efficient Archiving of Dynamic Linked Open Data

Javier D. Fernández, Axel Polleres, Jürgen Umbrich

Vienna University of Economics and Business, Vienna, Austria
{javier.fernandez,axel.polleres,juergen.umbrich}@wu.ac.at

Abstract. The Linked Data paradigm has enabled a huge shared infrastructure for connecting data from different domains which can be browsed and queried together as a huge knowledge base. However, structured interlinked datasets in this Web of data are not static but continuously evolving, which suggests the investigation of approaches to preserve Linked data across time. In this article, we survey and analyse current techniques addressing the problem of archiving different versions of semantic Web data, with a focus on their space efficiency, the retrieval functionality they serve, and the performance of such operations.

1 Introduction

The Linked Data paradigm promotes the use of the RDF [7] data model to publish structured data on the Web and to create data-to-data links between different data sources [9]. As a result, a continuously growing interconnected Web of data, consisting of typed hyperlinks between interconnected resources and documents has emerged over the past years and attracted the attention of several research areas, such as indexing, querying and reasoning over RDF data.

However, the emerging Web of data is not a static structure of linked datasets, but a dynamic framework continuously evolving; Distributedly and without notice, novel datasets are added, others are modified, abandoned to obsolescence or removed from the Web. All this without a centralized monitoring nor prefixed policy, following the scale-free nature of the Web. Applications and businesses leveraging the availability of certain data over time, and seeking to track data or conduct studies on the evolution of data, thus need to build their own infrastructures to preserve and query data over time.

Thus, preservation policies on Linked Open Data (LOD) collections emerge as a novel topic with the goal of assuring quality and traceability of datasets over time. However, previous experiences in traditional Web archives, such as the Internet Archive[1], with petabytes of archived information, already highlight scalability problems when managing evolving volumes of information at Web-scale, making the task of longitudinal query across time a formidable challenge with current tools [74]. It needs to be stressed that querying Web archives has to deal mainly with text, whereas structured interlinked data archiving shall focus

[1] http://archive.org.

on *structured queries* across time. In particular, several research challenges arise when representing and querying evolving structured interlinked data:

- How can we *represent archives* of continuously evolving linked datasets?
- How can we *minimize the redundant information* and *respect* the original modeling and provenance information of archives?
- How can emerging retrieval demands in archiving (*e.g.* time-traversing and traceability) be satisfied on such evolving interlinked data?
- How can certain time specific queries over archives be answered on existing technologies (*e.g.* SPARQL), providing the temporal validity of the returned bindings, and how can we capture the expresiveness of queries that cannot even be expressed in SPARQL itself (*e.g.* knowing if a dataset has changed, and how, in a certain time period)?

This work gains first insights into the problem of archiving and querying evolving semantic Web data. We first survey the most important related areas and techniques (Section 2). Then, we present current archiving models for dynamic Linked Open Data (Section 3), describing their main inherent characteristics and processing performance on common archiving retrieval needs. We conclude by pointing out current research challenges, in Section 4.

2 Related Work

Dynamic Linked Open Data. The use of RDF to expose Semantic Data on the Web has seen a dramatic increase over the last years. Several research areas have emerged aside (or resurfaced strongly), such as RDF indexing [24, 40] and optimized graph querying [62, 57] on the basis of SPARQL, ontological reasoning [13, 71, 49], and many others (schema mapping, graph visualization, etc.). Most of these areas and applications, though, consider a static snapshot of datasets, disregarding the dynamic nature of Web data, a lifecycle without order nor generally established policies. As an example of such behavior, LODStats[2], a project constantly monitoring the LOD, reports (in March 2015) 4,223 live datasets having almost 90 billion triples, in contrast to 5018 (54.3%) datasets which present problems for retrieving. Similar problems were previously reported by [69], and recently by the Dynamic Linked Data Observatory [32], which also provides the raw corpora, openly and continuously. Its latest studies found that 38% of 80K crawled Linked Data documents had changed in a seven-month window (the monitoring period). In particular, 23% of the total documents suffered from infrequent changes, and 8% were highly dynamic documents. Of the documents that changed, most updates affect values for RDF terms (27%), keeping the same number of triples, or just added triples (24%) which leads to monotonic additions. Regarding the availability of resources, 5% of documents disappeared permanently, whereas, on average, documents were unavailable 20% of the time.

All these results provide evidence that RDF datasets are rarely static, and tracking changes is receiving an increasing attention [67, 15, 72].

[2] http://stats.lod2.eu/.

Time modeling in RDF. Managing semantic archives succinctly stands for managing the time dimension in evolving RDF datasets. The time dimension is naturally present when representing information as it is probably one of the most important dimensions to locate events, actions and facts. There is, though, a complete catalog of definitions and theories about the temporal dimension [25, 1]; Time can be seen from multiple perspectives, from intervals (*10am to 5pm*) of validity of a fact to concrete time points (when an event has exactly happened), as well as durations (*1 hour, a decade*). The distinction between such perspectives is not always so clear or, better said, the temporal theories and managing tools tackle time under certain assumptions related to the final application. Additionally, temporal descriptions might be vague. For instance, TimeML [51], the ISO standard to annotate temporal information, allows to define recurrent events (*e.g. "twice a month"*) potentially without a concrete timepoint reference.

Temporal information has been discussed in temporal databases [58, 64, 30], XML documents [3, 53], and the World Wide Web [63, 2]. Although temporal representation and management in RDF and the Semantic Web have appeared recently, there is a growing interest in this area. See [11, 20] for a discussion on temporal aspects in the Semantic Web.

Research works in the Semantic Web area roughly distinguish three forms of annotating RDF/Linked Data with temporal information [54, 4]: (i) *document-centric* annotation, where time points are associated with whole RDF documents. This annotation can be implicit, for instance HTTP metadata can be used to detect changes [32], or explicit. In the latter case, specific vocabularies to annotate metadata about datasets, such as the Vocabulary of Interlinked Datasets (VoID)[3], can be used. In turn, RDF documents can be considered as digital objects following the path of other disciplines, such as Digital Libraries, with preservation standards for these objects, such as HDF[4] and PREMIS[5] (this latter including a semantic ontology for Linked Data compliance). Provenance information of data collections is also becoming a hot topic, given the distributed nature of Linked Open Data. In fact, the recent W3C PROV [22] standards can be used to annotate and query RDF documents with time [72].

Time can be also represented (ii) using *sentence-centric* annotation, explicitly defining the temporal validity, whether a time point or intervals, at the level of statements/triples [68, 65, 50, 23, 78], and (iii) in a *relationship-centric* model, encapsulating time into n-ary relations [43]. In this latter, a specific resource identifies the time relation, and make use of it to link other related resources [75, 39]. This can be seen as a particular case of multidimensional modeling [33, 17]. In [56], the authors study different general time modeling behaviors stating that n-ary is the most used for experts.

As yet another practical approach which does not fall into one of these strict categories, in [27] the authors build a Knowledge Base of facts (Yago2) from

[3] http://www.w3.org/TR/void/.
[4] http://www.hdfgroup.org/products/hdf4/.
[5] http://www.loc.gov/standards/premis/.

Wikipedia, and enhance it with temporal and spatial dimension: they distinguish between the temporal dimension of entities (resources) and facts (sentences). They develop a method to capture time and space from facts, as well as rules for propagating such information to other related facts. To do so, they consider static/permanent relations (*e.g.* ISBN, name), creation relations (*e.g.* paint, create), destruction relation (*e.g.* died, destroy), and rules created on top. Most important for preservation, each fact is assigned a time point denoting its extraction and insertion into the Knowledge Base in order to capture provenance information and allow systems to select facts from certain points of time. This latter corresponds to the *transaction time* in the literature on temporal databases [58], *i.e.* the period of time during with a fact is present in the system, in contrast to the *valid time*, *i.e.* the data entered by the user of when a fact is true in reality.

Finally, the OWL-Time[6] ontology provides an RDF vocabulary to model – but not to directly process and query – such temporal aspects.

Structured query languages managing time. Within the database community, several temporal query languages have been designed on the basis of SQL-like modifications, such as TQuel [59], or the TSQL2 language [60] designed for temporal relational databases [58]. The main time features of these proposals are based on defining operations between time intervals [25, 1], such as computing *overlapping* and *meeting* points of two intervals, or *before*, *equal*, *during* and *finish* relationships between them.

Structured query languages managing time represented in RDF mainly follow this approach. T-SPARQL [19] is a temporal extension of SPARQL, motivating the proposal on the need of keeping track of the changes in the legal domain. The T-SPARQL language assumes that the temporal information is represented in RDF using a sentence-centric annotation where timestamps are associated with RDF triples. Then, it extends SPARQL embedding aforementioned temporal features of the TSQL2 language [60]. In contrast, the author does not provide a feasible implementation, but names two special index structures which allow for efficient processing of temporal queries: tGRIN [50] and keyTree [65].

The tGRIN proposal is based on triples annotated with time (tRDF), *i.e.* a sentence annotation modeling, but they consider time annotations in the edges of the relations. Then, they define a variation of SPARQL augmented with these temporal annotations on the edges (either variable or constant), referred to as tRDF queries. Finally, they build a balance tree keeping together those nodes with a "close" timing (as they should be queried together). Nevertheless, both the construction and the query can result in costly operations.

SPARQL-ST [48] is a SPARQL extension supporting spatio-temporal queries, limited to special FILTER conditions, also on the basis of sentence annotation. Regarding the query language in [27], authors make use of reification, as each fact has an identifier and time points are associated to facts. However, searching in a SPARQL fashion produces a large number of joins which is not easy for

[6] http://www.w3.org/TR/owl-time/.

non-experts. Thus, they designed the so-called SPOTL view, a syntactic sugar modification of SPARQL to avoid such joins when specifying time or location. In particular, four types of time predicates can be added after each (s,p,o) pattern: *overlaps, during, before* and *after*. This allows to query interesting events with few constructions. Based on a similar idea, stSPARQL [8] extends SPARQL with minimal constructions to query valid times of linked geospatial data.

In turn, AnQL [78] is a SPARQL extension focused on querying annotated graphs. These annotations can refer to several domains, such as trust or fuzzy values, provenance and temporal information (*e.g.* temporal validity), for which the proposal highlights some specific issues and extensions to cope with multiple domains. Finally, other research efforts focus on representing and querying temporal information in the Web Ontology Language [26] (OWL) [46, 6].

Web archives. Similar scenarios were recently envisioned for Web archiving[7], *i.e.* maintaining snapshots of the Web at different times. In this respect, active non-profit organizations, such as Common Crawl[8] and the Internet Memory[9], provide open Web crawl data which can be used for third parties. In particular, the Web Data Commons project[10] has extracted the hyperlink graph[11] from the data from Common Crawl, also providing it for open use.

However, current Web archiving offers limited query services, mostly restricted to dereferencing URLs across certain time points. Note that the Web *archive* size will increase several orders of magnitude across time: a system that should index and provide full-text queries to such an archive will need to deal with amounts of information which largely surpass the volume managed by leading Web search engines. Although several works identify this challenge of Web archive search [12, 74, 18] no satisfying solution exists to date. In fact, this can be seen as one on the biggest Big Data problems [18].

Other issues & Discussion. Many semantic Web applications currently work mainly with the most recent snapshot of RDF data collections from the Web. Acknowledging that this could cope the requirements of some information services, more advanced requirements, such as longitudinal querying over time and queries regarding of the evolution of data, cannot be achieved. Consider, for instance, the open data portals of certain governments. These portals usually have several contributors from different governmental agencies, or even try to provide a unique access point to information from diverse governmental levels. Recent projects monitoring Open Data portals, such as the Open Data Portal Watch [70] and the Open Data Monitor[12], reinforce the idea that these portals are growing unconstrained. Thus, there is an ongoing need to archive information whether locally at the site of each contributor itself, or centralized. In any case,

[7] Up-to-date bibliography at http://digital.library.unt.edu/ark:/67531/metadc172362.
[8] http://commoncrawl.org/.
[9] http://internetmemory.org/en/.
[10] http://webdatacommons.org/.
[11] Latest graph (April 2014) covers 1.7 billion web pages and 64 billion hyperlynks.
[12] http://www.opendatamonitor.eu/.

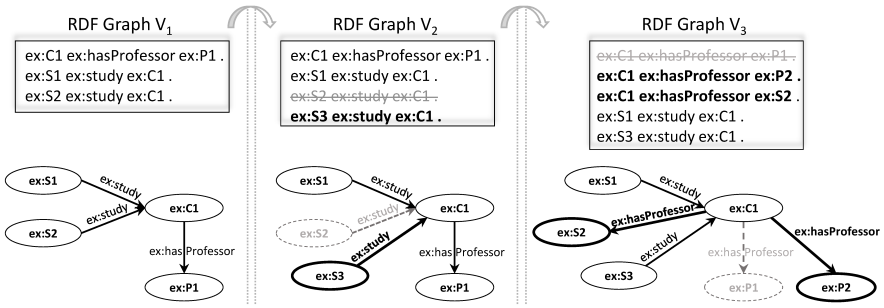

Fig. 1. Example of RDF graph versions.

it could be relevant for analysis purposes to know the evolution of a collection over time. For external applications and services built on top of such collections, a searchable record of changes should be useful to integrating such information accordingly. In this sense, RDF graph replication [29, 52, 55, 66] addresses the slightly different problem of mirroring complete or partial graphs (locally or in different nodes), and to propagate changes between the subsets. Although in this case the main objective is to optimize the synchronization, some tasks, such as semantic differences computation and size reduction, can be seen as common issues both for data replication and LOD archiving.

Finally, what was stated in a dataset *at a certain time point* in the past may be taken into account for legals aspects in the same way its used in Web archiving [28]. Data journalism is another particular area in which tracking and comparing information over time is especially relevant.

3 Modelling Archives of Dynamic Linked Open Data

In this section we present current practical approaches to archive dynamic Linked Open Data. We then discuss the desired retrieval functionality and how different models specifically tackle such query demands. Our use case is depicted in Figure 1, which shows a sequence of three snapshots for an RDF graph. In this example, the original version V_1 models the information on two students *ex:S1* and *ex:S2* studying in a course *ex:C1*, whose professor is *ex:P1*. In the second version V_2, *ex:S2* disappeared in favour of a new student *ex:S3*. Finally, in version V_3, professor *ex:P1* leaves the course to a pair of professors: a new professor *ex:P2* and the former *ex:S2* who reappears under a different role.

3.1 Archiving policies

Several research efforts address the challenge of archiving the increasingly dynamic data exposed as Linked Data. The main related works make use of three storage strategies (slightly adapted from [67]), namely *independent copies (IC)*, *change-based (CB)* and *timestamp-based (TB)* approaches. Figure 2 shows different archiving policies for our running example.

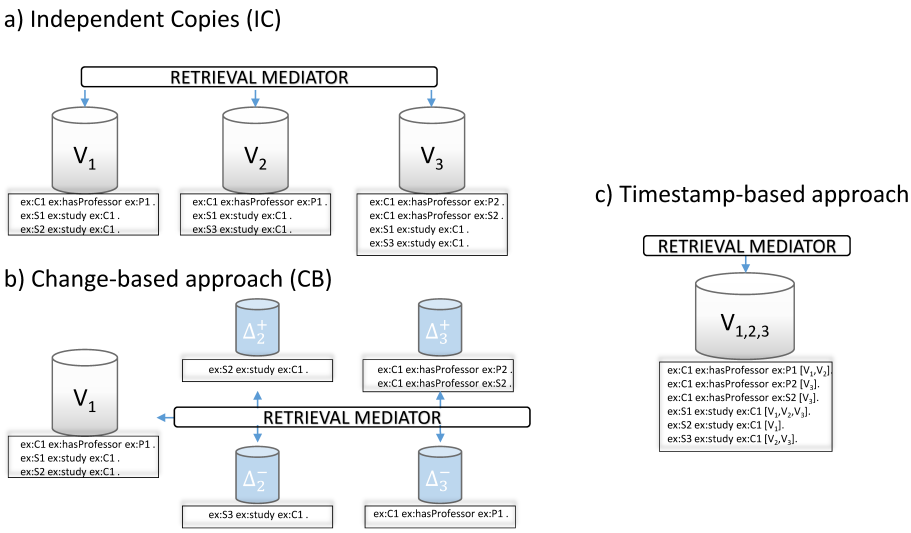

Fig. 2. Common archiving policies on RDF graph versions.

Independent Copies (IC). This basic policy [34, 45] stores and manages each version as a different, isolated dataset, as shown in Figure 2 (a). Nonetheless, a metadata characterization could be built on top in order to provide a catalogue of the different available versions, e.g. using the Data Catalog Vocabulary (DCAT) [38] and the Provenance Ontology (PROV) [22].

IC suffers from well-known scalability problems, as the static core (triples that do not change), is fully repeated across versions.

Change-based approach (CB). This policy addresses the previous scalability issue by computing and storing the differences (deltas) between versions, as can be seen in Figure 2 (b). In this particular example, differences are computed at the level of triples (*low-level* deltas [73, 15, 77]), distinguishing between added (Δ^+) and deleted (Δ^-) triples. Change detection is thus based on a basic *language of changes* describing the change operations, typically marking added and deleted triples, which can be shared in markup languages such as RDF Patch[13]. Complementary recent works focus on computing dataset differences in a distributed way, such as rdfDiff[14] and G-Diff [31], both working on MapReduce.

Other approaches works on extracting human-readable (*high-level* deltas [47, 44]) with the purpose of obtaining a more concise explication on the whys and hows of the changes (*e.g.* deltas can state that a class has been renamed, and this affects all the instances). On the one hand, *low-level* deltas are easy to detect and manage, and applies to any valid RDF dataset. On the other hand, *high-level* deltas are more descriptive and can be more concise, but this is at the cost of relying on an underlying semantics (such as RDFS or OWL), and they are more complex to detect and manage [76].

[13] http://afs.github.io/rdf-patch/.
[14] https://github.com/paulhoule/infovore/wiki/rdfDiff.

Timestamp-based approach (TB). This approach can be seen as a particular case of the aforementioned *sentence-centric* annotation to model the time in RDF [68, 65, 50, 23, 78]. Instead of explicitly defining the temporal validity of statements/triples, in the LOD archiving, each sentence locally holds the timestamp of the version. Again, the static core of the archive would produce repetitions, as static triples would be labelled with all the present timestamps. In order to save space avoiding such repetitions, practical proposals annotate the triples only when they are added or deleted. That is, the triples are augmented by two different fields: the created and deleted (if present) timestamps [41, 72].

3.2 Retrieval Functionality

Querying evolving semantic Web data is an emerging but challenging topic. Considering several aspects from previous categorizations [61, 35], we can distinguish six different types of retrieval needs grouped under several dimensions, shown in Table 1. The classification regards the query type (materialization or structured query) and the main focus (version/delta) of the involved query. We also distinguish the time (single/cross-time queries) for the structured queries.

Version materialization: It involves to retrieve a given version, or the closest version/s from a certain given period.

Although this can be seen as a straightforward, basic demand, in fact (i) it is the most common feature provided by large scale archives, such as current Web archiving (see Section 2) that mostly dereferences URLs across certain time points, and (ii) it still presents a challenge at Web scale, as size will increase several orders of magnitude across time. This functionality is also intensively used in related areas, such as retrieving a certain state in revision control systems.

Single-version structured queries: Structured queries must be satisfied on a specific version, typically, but not limited to, the newest one.

In this case, the retrieval demands are aligned with current state-of-the-art query resolution over semantic data. That is, one could expect to work on an RDF archive in a similar way than to any RDF store that serves, for instance SPARQL resolution. Likewise, the same complexities applies, with the added difficulty of switching between versions.

For instance, in our running example from Figure 1, one could ask *what lecture was given by certain teacher at a certain time*, or if *two given students attended the same course in a given time.*

Cross-version structured queries: Structured queries must be satisfied across different versions.

This feature opens up the possibility to resolve time-traversal queries, thus coping with information needs dealing with evolution patterns. This evolution can be tracked at two different levels: (i) a simple triple/sentence pattern, for instance, in our running example one may be interested in *knowing all the courses attending by a certain student*, and (ii) a complex graph pattern query, such as *retrieving those subjects who have played the role of student and teacher of the same course.* Both cases are related to previous work on structured query

Type / Focus	Materialization	Structured Queries: Single time	Structured Queries: Cross time
Version	Version Materialization *-get snapshot at time t_i*	Single-version structured queries *-lectures given by certain teacher at time t_i*	Cross-version structured queries *-subjects who have played the role of student and teacher of the same course*
Delta	Delta Materialization *-get delta at time t_i*	Single-delta structured queries *-students leaving a course between two consecutive snapshots, i.e. between t_{i-1}, t_i*	Cross-delta structured queries *-evolution of added/deleted students across versions*

Table 1. Classification and examples of retrieval needs.

RETRIEVAL NEED	POLICIES: Indep. Copies (IC)	Change-based (CB)	Timestamp-based (TB)
Version Materialization	Low	Medium	Medium
Delta Materialization	Medium	Low	Low
Single-version structured queries	Medium	Medium	Medium
Cross-version structured queries	High	High	Medium
Single-delta structured queries	High	Medium	Medium
Cross-delta structured queries	High	High	Medium

Table 2. Processing of retrieval needs (level of complexity).

languages managing time (see Section 2). Nonetheless, the first one requires less expressiveness, and thus it could be provided by simpler resolution mechanisms.

Delta materialization: In this case, the main focus of the retrieval need is the changes between two or more given versions.

While this functionality may seem underexploited in current LOD archiving, increasingly LOD adoption would also bring RDF authoring to be widespread and distributively performed. In this scenario, version difference and its related operations (merge, conflict resolution, etc.) would be as crucial as they are in revision control systems. Besides authoring, third-party systems relying on such evolving datasets also might need to maintain a fresh version and thus updating policies can be based on knowing and materializing version differences.

Single-delta structured queries: Structured queries must be satisfied on a specific change instance of the dataset.

This information demand particularly focuses on the differences between two versions, which are typically but not always consecutive. Besides other evolution studies, these queries play a main role in monitoring and maintenance processes. For instance, in our running example, one could be interested in *knowing the students leaving a course between two versions*. In general, it is interesting to know if certain addition/deletion or modification pattern applies, as this could trigger other actions. This monitoring can impact on related areas such as view maintenance, schema mappings and inference recomputation.

Cross-delta structured queries: Structured queries must be satisfied across the differences of the dataset, thus allowing for fully fledged evolution studies.

In this case, the retrieval needs could be seen (i) as a generalization of the previous scenario, thus fostering the evolution studies and feeding the monitoring and maintenance processes, and (ii) as a particular realization of cross-version structured queries. Nonetheless, in cross-delta structured queries we put the focus on knowing the differences, i.e. the hows of the evolution process. For instance, complex queries on our example could require to *know the evolution in the number of added/deleted students or teachers across several version.*

3.3 Retrieval Processing

Finally, we briefly discuss how the presented archiving models can tackle the aforementioned retrieval needs. First of all, note that all models represent the same information, in complementary ways, so that all retrieval needs could be theoretically satisfied. However, their aims clearly differ, thus they could present important drawbacks. Table 2 summarizes the qualitative level of complexity (low, medium, high) required to satisfy each type of retrieval demand.

Independent Copies (IC) may suffer from scalability problem in space, but it is a straightforward, simple approach that could fit for basic retrieval purposes, as version materialization, with low effort. In fact, this approach is widely used to directly provide historical version dumps (typically compressed to reduce space needs of textual RDF formats), such as in DBpedia[15] and other projects serving Linked Open Data snapshots, such as the dynamic Linked Data Observatory[16]. In contrast, the rest of operations requires medium or high processing efforts. A potential retrieval mediator (depicted in Figure 2 (a)) should be placed on top of the versions, with the challenging tasks of i) computing deltas at query time to satisfy delta-focused queries, ii) loading/accessing the appropriate version/s and solve the structured queries, and iii) performing both previous tasks for the particular case of structured queries dealing with deltas.

Change-based approaches (CB) reduce the required space but at the cost of requiring additional computational costs for delta propagation and thus version-focused retrieving operations. In general, as shown in Figure 2 (b), a query mediator should access a materialized version and the subsequent deltas. In this case, delta materialization is obviously costless but i) it requires of the aforementioned delta propagation to solve version-focused queries and ii) to load/access the appropriate delta/s or re-created version/s for structured queries.

Nonetheless, note that his strategy is highly configurable, both in (a) the aforementioned mechanism to detect and store the differences (e.g. low/high level deltas), (b) whether to apply direct deltas (computing the changes of version V_i with respect to version V_{i-1}) or reverse deltas (computing the changes of version V_{i-1} with respect to version V_i) and (c) whether to store all subsequence deltas or store the full version materialization in some intermediate steps. Recent works inspect the latter tradeoffs. On the one hand, [15] precomputes an aggregation of all deltas, so that it improves cross-delta computation at the cost of augmenting space overheads. On the other hand, [61] proposes a theoretical cost model to adopt a hybrid (IC+CB) approach. These costs highly depend on the difficulties of constructing and reconstructing versions and deltas, which may depend on multiple and variable factors. Another intermediate solution [67] builds a partial order index keeping a hierarchical track of changes. This proposal, though, is a limited variation of delta computation and it is only tested with datasets having some thousand triples. Same scalability issues applies for a hypergraph-based solution [14], storing the information of version in hyperedges.

[15] http://wiki.dbpedia.org/Downloads
[16] http://swse.deri.org/dyldo/data/

Timestamp-based approaches (TB) conceptually manage one augmented graph containing all versions, which are labelled accordingly. As stated, most practical approaches, such as [41, 72, 21], annotate the insertion/deletion point of each triple, thus saving space. These approaches manage versions/deltas under named/virtual graphs, so that the retrieval mediator (depicted in Figure 2 (c)) can rely on existing solutions providing named/virtual graphs. In Table 2 we consider these practical cases and thus we report that, except for delta materialization, all retrieval demands can be satisfied with medium processing efforts given that i) version materialization requires to rebuild the delta similarly to CB, and ii) structured queries may need to skip irrelevant triples [41].

4 Discussion

Dynamic Linked Open Data archiving is a relevant and emerging area of interest [5] which has its roots in Web archives where, unfortunately, the few current approaches are seriously compromised by scalability drawbacks at Web scale [12]. In addition, these proposals include basic search capabilities, whereas structured and time-traversing queries also constitute emerging retrieval demands [74].

Specific versioning, data replicas and archiving of interlinked data are still in an early stage of research [72, 15, 67], while none of the analysed representations have been neither designed nor applied at the scale of Linked Open Data.

Our current efforts to foster efficient archiving of dynamic Linked Open Data focus on two complementary challenges. On the one hand, improving scalability of archives, which involves to manage them on a modular, distributed fashion, while reducing the high levels of verbosity/redundancy. On the other hand, optimizing query resolution, specially for those retrieval demands that requires to scale up to large volumes of data, along different dimensions.

Compression and Indexing of Archives. One promising way of managing such collections at large scale is to take advantage of the information redundancy to minimize its representation through compression and to provide indexing and thus query resolution on the compressed information. Compressing and indexing highly repetitive text collections is an active research area. Grammar-based compressors [42] infer a grammar which generates the given text, hence they are particularly suitable for texts comprising many repeated substrings because these can be effectively encoded through the grammar rules. In addition, latest proposals enable direct access to the data [10]. Similar goals have been pursued on the basis of the Lempel-Ziv LZ78 [80] or LZ77 [79] variants, with their counterpart searchable proposals [36, 37]. Most of the approaches allowing direct access assume that the information of which texts are close variants of which can be identified. Thus, representative baseline texts can be selected while the related texts can be compressed referencing their representatives [10].

Although archiving dynamic Linked Open Data has also to tackle text redundancy, its distinguishing feature is the presence of a semantic structure. In this respect, specific RDF compression [16] emerges as an ideal solution to achieve highly-compressed representations of RDF archives at large scale.

Query resolution of Archives. Structured query mechanisms for temporal data are mainly based on traditional relational proposals. While some work has been done on structured query languages managing time in RDF [19, 48, 27], none of the proposals is specific for archiving evolving interlinked data. In turn, there is still a large interest in scalable RDF indexing [24, 40] and query optimization [62, 57], whose performance is critical when managing very large datasets.

An efficient solution is thus a formidable challenge, which should consider a scalable model for archiving, efficient compression and indexing methods that supports an expressive temporal query language, which, all together, will enable to gain novel insights from Linked Open Data across time.

Acknowledgments

Javier D. Fernández is funded by Austrian Science Fund (FWF): M1720-G11.

References

1. J. F. Allen. Towards a General Theory of Action and Time. *Artificial intelligence*, 23(2):123–154, 1984.
2. O. Alonso, J. Strötgen, R. A Baeza-Yates, and M. Gertz. Temporal information retrieval: Challenges and opportunities. In *Proc. of TempWeb*, volume CEUR-WS 707, paper 1, 2011.
3. T. Amagasa, M. Yoshikawa, and S. Uemura. A Data Model for Temporal XML Documents. In *Proc. of DEXA*, pp. 334–344, 2000.
4. A. Analyti and I. Pachoulakis. A Survey on Models and Query Languages for Temporally Annotated RDF. *International Journal of Advanced Computer Science and Applications*, 3(9):28–35, 2012.
5. S. Auer, T. Dalamagas, H. Parkinson, F. Bancilhon, G. Flouris, D. Sacharidis, P. Buneman, D. Kotzinos, Y. Stavrakas, V. Christophides, G. Papastefanatos, and K. Thiveos. Diachronic Linked Data: Towards Long-term Preservation of Structured Interrelated Information. In *Proc. of WOD*, pp. 31–39, 2012.
6. S. Batsakis, K. Stravoskoufos, and E. G. M. Petrakis. Temporal Reasoning for Supporting Temporal Queries in OWL 2.0. In *Proc. of KES*, pp. 558–567. 2011.
7. D. Beckett. *RDF/XML Syntax Specification (Revised).* W3C Recom. 2004.
8. K. Bereta, P. Smeros, and M. Koubarakis. Representation and Querying of Valid Time of Triples in Linked Geospatial Data. In *Proc. of ESWC*, pp. 259–274. 2013.
9. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5:1–22, 2009.
10. F. Claude and G. Navarro. Self-Indexed Text Compression using Straight-Line Programs. In *Proc. of MFCS*, pp. 235–246. 2009.
11. G. Correndo, M. Salvadores, I. Millard, and N. Shadbolt. Linked Timelines: Temporal Representation and Management in Linked Data. In *Proc. of COLD*, volume CEUR-WS 665, paper 7. 2010.
12. M. Costa, D. Gomes, F. Couto, and M. Silva. A Survey of Web Archive Search Architectures. In *Proc. of WWW Companion*, pp. 1045–1050, 2013.
13. J. De Bruijn and S. Heymans. Logical foundations of (e)RDF(S): Complexity and reasoning. In *Proc. of ISWC*, pp. 86–99. 2007.

14. I. Dong-Hyuk, Z. Nansu, K. Eung-Hee, Y. Seokchan, and K. Hong-Gee. A Hypergraph-based Storage Policy for RDF Version Management System. In *Proc. of ICUIMC*, pp. 74:1–74:5, 2012.
15. I. Dong-Hyuk, L. Sang-Won, and K. Hyoung-Joo. A Version Management Framework for RDF Triple Stores. *International Journal of Software Engineering and Knowledge Engineering*, 22(1):85–106, 2012.
16. J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF Representation for Publication and Exchange (HDT). *Journal of Web Semantics*, 19:22–41, 2013.
17. M. Gergatsoulis and P. Lilis. Multidimensional RDF. In *Proc. of OTM*, pp. 1188–1205. 2005.
18. D. Gomes, M. Costa, D. Cruz, J. Miranda, and S. Fontes. Creating a Billion-scale Searchable Web Archive. In *Proc. of WWW Companion*, pp. 1059–1066, 2013.
19. F. Grandi. T-SPARQL: A TSQL2-like Temporal Query Language for RDF. In *Proc. of ADBIS*, pp. 21–30. 2010.
20. F. Grandi. Introducing an Annotated Bibliography on Temporal and Evolution Aspects in the Semantic Web. *SIGMOD Rec.*, 41(4):18–21, 2013.
21. M. Graube, S. Hensel, and L. Urbas. R43ples: Revisions for triples. In *Proc. of LDQ*, volume CEUR-WS 1215, paper 3, 2014.
22. P. Groth and L. Moreau. *PROV-overview: an Overview of the PROV Family of Documents*. W3C Working Group Note. 2013.
23. C. Gutierrez, C.A. Hurtado, and A. Vaisman. Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):207–218, 2007.
24. A. Harth, J. Umbrich, A. Hogan, and S. Decker. YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In *Proc. of ISWC*, pp. 211–224. 2007.
25. P. Hayes. A Catalog of Temporal Theories. *Tech. Report UIUC-BI-AI-96-01*, 1995.
26. P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. *OWL 2 Web Ontology Language Primer (Second Edition)*. W3C Recom. 2012.
27. J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence*, 194:28–61, 2013.
28. B. A Howell. Proving Web History: How to Use the Internet Archive. *Journal of Internet Law*, 9(8):3–9, 2006.
29. L. D. Ibáñez, H. Skaf-Molli, P. Molli, and O. Corby. Live linked data: Synchronising semantic stores with commutative replicated data types. *International Journal of Metadata, Semantics and Ontologies*, 8(2):119–133, 2013.
30. C. S. Jensen, C. E. Dyreson, M. Böhlen, et al. The consensus Glossary of Temporal Database Concepts-February 1998 version. *Temporal Databases: Research and Practice*, pp. 367–405, 1998.
31. A. Jinhyun, I. Dong-Hyuk, E. Jae-Hong, Z. Nansu, and K. Hong-Gee. G-Diff: A Grouping Algorithm for RDF Change Detection on MapReduce. In *Proc. of JIST*, pp. 230–235. 2015.
32. T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne, and A. Hogan. Observing Linked Data Dynamics. In *Proc. of ESWC*, pp. 213–227. 2013.
33. B. Kampgen. *Flexible Integration and Efficient Analysis of Multidimensional Datasets from the Web*. PhD thesis, Karlsruhe Institute of Technology, 2015.
34. M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology versioning and change detection on the web. In *Proc. of EKAW*, pp. 197–212. 2002.
35. G. Koloniari and E. Souravlias, D.and Pitoura. On Graph Deltas for Historical Queries. In *Proc. of WOSS*, volume arXiv:1302.5549, 2012.

36. S. Kreft and G. Navarro. On compressing and indexing repetitive sequences. *Theoretical Computer Science*, 483:115–133, 2013.
37. S. Kuruppu, S. J Puglisi, and J. Zobel. Relative Lempel-Ziv Compression of Genomes for Large-Scale Storage and Retrieval. In *Proc. of SPIRE*, pp. 201–206. 2010.
38. F. Maali, J. Erickson, and P. Archer. *Data catalog vocabulary (DCAT).* W3C Recom. 2014.
39. V. Milea, F. Frasincar, and U. Kaymak. Knowledge Engineering in a Temporal Semantic Web Context. In *Proc. of ICWE*, pp. 65–74, 2008.
40. T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19:91–113, 2010.
41. T. Neumann and G. Weikum. x-RDF-3X: Fast querying, high update rates, and consistency for RDF databases. *Proc. of VLDB Endowment*, 3(1-2):256–263, 2010.
42. C. G. Nevill-Manning, I. H. Witten, and D. L. Maulsby. Compression by induction of hierarchical grammars. In *Proc. of DCC*, pp. 244–253, 1994.
43. N. Noy, A. Rector, P. Hayes, and C. Welty. Defining N-ary Relations on the Semantic Web. *W3C Working Group Note*, 2006.
44. N. F. Noy and M. A. Musen. Promptdiff: A Fixed-Point Algorithm for Comparing Ontology Versions. In *Proc. of IAAI*, pp. 744–750. 2002.
45. N. F. Noy and M. A. Musen. Ontology Versioning in an Ontology Management Framework. *IEEE Intelligent Systems*, 19(4):6–13, 2004.
46. M. J. O'Connor and A. K. Das. A Method for Representing and Querying Temporal Information in OWL. In *Proc. of BIOSTEC*, pp. 97–110, 2011.
47. V. Papavasileiou, G. Flouris, I. Fundulaki, D. Kotzinos, and V. Christophides. High-level Change Detection in RDF(S) KBs. *ACM Trans. Database Syst.*, 38(1), 2013.
48. M. Perry, P. Jain, and A. P. Sheth. SPARQL-ST: Extending SPARQL to Support Spatiotemporal Queries. *Geospatial Semantics and the Semantic Web*, 12:61–86, 2011.
49. A. Polleres, A. Hogan, R. Delbru, and J. Umbrich. RDFS and OWL Reasoning for Linked Data. In *Proc. of RW*, pp. 91–149. 2013.
50. A. Pugliese, O. Udrea, and V. S. Subrahmanian. Scaling RDF with time. In *Proc. of WWW*, pp. 605–614, 2008.
51. J. Pustejovsky, J. M. Castaño, R. Ingria, R. Sauri, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proc. of IWCS*, 2003.
52. L. Rietveld. Replication for Linked Data. In *Proc. of ISWC*, pp. 415–423. 2012.
53. F. Rizzolo and A. A. Vaisman. Temporal XML: Modeling, Indexing, and Query Processing. *The VLDB Journal*, 17(5):1179–1212, 2008.
54. A. Rula, M. Palmonari, A. Harth, S. Stadtmüller, and A. Maurino. On the Diversity and Availability of Temporal Information in Linked Open Data. In *Proc. of ISWC*, pp. 492–507. 2012.
55. B. Schandl. Replication and Versioning of Partial RDF Graphs. In *Proc. of ESWC*, pp. 31–45. 2010.
56. A. Scheuermann, E. Motta, P. Mulholland, A. Gangemi, and V. Presutti. An Empirical Perspective on Representing Time. In *Proc. of K-CAP*, pp. 89–96, 2013.
57. M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL Query Optimization. In *Proc. of ICDT*, pp. 4–33, 2010.
58. R. T. Snodgrass. Temporal Databases. *IEEE Computer*, 19:35–42, 1986.
59. R. T. Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems (TODS)*, 12(2):247–298, 1987.

60. R. T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.
61. K. Stefanidis, I. Chrysakis, and G. Flouris. On Designing Archiving Policies for Evolving RDF Datasets on the Web. In *Proc. of ER*, pp. 43–56. 2014.
62. M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds. SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation. In *Proc. of WWW*, pp. 595–604, 2008.
63. J. Strötgen, O. Alonso, and M. Gertz. Identification of Top Relevant Temporal Expressions in Documents. In *Prof. of TempWeb*, pp. 33–40, 2012.
64. A. U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation*. 1993.
65. J. Tappolet and A. Bernstein. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In *Proc. of ESWC*, pp. 308–322. 2009.
66. G. Tummarello, C. Morbidoni, R. Bachmann-Gmür, and O. Erling. RDFSync: Efficient Remote Synchronization of RDF Models. In *Proc. of ISWC*, pp. 537–551. 2007.
67. Y. Tzitzikas, Y. Theoharis, and D. Andreou. On Storage Policies for Semantic Web Repositories That Support Versioning. In *Proc. of ESWC*, pp. 705–719. 2008.
68. O. Udrea, D. R. Recupero, and V. S. Annotated RDF. *ACM Transactions on Computational Logic (TOCL)*, 11(2):1–41, 2010.
69. 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*, 2010.
70. J. Umbrich, S. Neumaier, and A. Polleres. Towards assessing the quality evolution of Open Data portals. In *Proc. of ODQ*, 2015.
71. J. Urbani, S. Kotoulas, E. Oren, and F. Van Harmelen. Scalable Distributed Reasoning using Mapreduce. In *Proc. of ISWC*, pp. 634–649. 2009.
72. M. Vander Sander, P. Colpaert, R. Verborgh, S. Coppens, E. Mannens, and R. Van de Walle. R&Wbase: Git for triples. In *Proc. of LDOW*, volume CEUR-WS 996, paper 1, 2013.
73. M. Volkel, W. Winkler, Y. Sure, S.R. Kruk, and M. Synak. Semversion: A versioning system for rdf and ontologies. In *Proc. of ESWC*, 2005.
74. G. Weikum, N. Ntarmos, M. Spaniol, P. Triantafillou, A. Benczúr, S. Kirkpatrick, P. Rigaux, and M. Williamson. Longitudinal Analytics on Web Archive Data: It's About Time! In *Proc. of CIDR*, pp. 199–202, 2011.
75. C. Welty, R. Fikes, and S. Makarios. A reusable ontology for fluents in owl. In *Proc. of FOIS*, pp. 226–236, 2006.
76. F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, and M. Sabou. Ontology evolution: a process-centric survey. *The Knowledge Engineering Review*, 30(01):45–75, 2015.
77. D. Zeginis, Y. Tzitzikas, and V. Christophides. On Computing Deltas of RDF/S Knowledge Bases. *ACM Transactions on the Web (TWEB)*, 5(3):14, 2011.
78. A. Zimmermann, N. Lopes, A. Polleres, and U. Straccia. A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12:72–95, 2012.
79. J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.
80. J. Ziv and A. Lempel. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.