Predicting SPARQL Query Dynamics

Alberto Moya Loustaunau
IMFD; DCC, University of Chile
Supervised by: Aidan Hogan

ABSTRACT

The SPARQL language is the recommendation for querying Linked Data, but querying SPARQL endpoints has problems with performance, particularly when clients remotely query SPARQL endpoints over the Web. Traditionally, caching techniques have been used to deal with performance issues by allowing the reuse of intermediate data and results across different queries. However, the resources in Linked Data represent real-world things which change over time. The resources described by these datasets are thus continuously created, moved, deleted, linked, and unlinked, which may lead to stale data in caches. This situation is more critical in the case of applications that consume or interact intensively with Linked Data through SPARQL, including query engines and browsers that constantly send expensive and repetitive queries. Applications that leverage Linked Data could benefit from knowledge about the dynamics of changing query results to efficiently deliver accurate services, since they could refresh at least the dynamic part of the queries. Along these lines, we want to address open questions in terms of assessing the dynamics of SPARQL query results in order to improve the way applications access dynamic Linked Data, making queries more efficient and ensuring fresher results.

CCS CONCEPTS

• **Information systems** → **Temporal data**; **Resource Description Framework (RDF)**; **Query languages**.

KEYWORDS

Dynamics, Linked Data, SPARQL, RDF

ACM Reference Format:
Alberto Moya Loustaunau. 2022. Predicting SPARQL Query Dynamics. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487553.3524195

1 INTRODUCTION & PROBLEM

The Linked Data (LD) principles[1] provide a flexible publishing paradigm to integrate and interlink data on the Web. Following the LD principles, a lot of organizations are publishing valuable information online using the Resource Description Framework (RDF) and other LD technologies. The standardized approach to query these kinds of data is SPARQL, the recommendation of the W3C[2] to query RDF. There are many public SPARQL endpoints that allow any data consumer to query RDF datasets on the Web [35]. Since the LD principles have been created, until now, the Linked Open Data (LOD) Cloud[3] has grown significantly [27]. Additionally, a wide variety of applications use Linked Data sourced from the Web to offer services.

Currently, querying SPARQL Endpoints has problems like network instability and latency, which affect query efficiency [2]. Many research efforts have been dedicated to circumvent this problem [37] and caching is one of the popular directions. While most research efforts focus on providing a server-side caching mechanism, being embedded in triple stores, client-side caching has not been fully explored. The distributed Web-based nature of the data generally requires LOD applications to keep local copies of the data in a cache for faster access at runtime. This situation is more critical in the case of applications that consume or interact intensively with Linked Data through the SPARQL Query Language, such as query engines and browsers that may also constantly send expensive and repetitive queries.

The main problem of client-side caching is the dynamic nature of the data on the Web: as the data on the Web changes, these caches or indexes no longer reflect the current state of the data anymore and need to be updated, but this is a major challenge, because the remote sources are updated autonomously and independently of the local copies. By caching the answers of queries and using the cached replies whenever possible, client-side caches reduce the network traffic between clients and servers, the load on servers, and the average user-perceived latency of query processing. However, for caches to be useful, cache consistency must be maintained, that is, cached copies should be updated when the data that gave rise to them change. Ensuring a strong consistency in which no stale data will ever be returned to the user could be expensive. Thus, several weakly consistent approaches have been proposed that try to keep the local data of the applications updated at lower cost [6, 14, 23].

Many applications that consume Linked Data face challenges related to changes in the underlying data, whose key goal is keeping data fresh while keeping response times low. Also, many investigations [3, 8, 13, 27, 29] have shown that the data published and interlinked on the Web is subject to frequent changes. The volume and velocity of changes constitute important challenges that applications must handle to keep results up-to-date. Since datasets change over time, long-running applications that cache and repeatedly use query results obtained from a SPARQL endpoint may resubmit the queries regularly to ensure up-to-dateness. As a result, applications often get the same results in several query executions, resulting in an unnecessary computation or if using a cache, they

[1] https://www.w3.org/DesignIssues/LinkedData.html

[2] https://www.w3.org/
[3] https://lod-cloud.net/

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524195

may often give the user stale results when the backend data change between re-executions.

Some works have proposed to learn from historical query result evolution [14, 29, 34]. However, considering the historical evolution of all results for a query is expensive because it is necessary to have the previous complete versions of data and to obtain the results of the execution of the query in each version. We would like to explore the dynamics of Linked Datasets at the level of queries with the intention of finding patterns that allow us to model the dynamics of SPARQL query results without relying on a history of results.

Recent works have found some general correlations with datasets dynamics, such as the domains, predicates, and schema associated with the data [13, 23, 29]. Based on such correlations, hybrid approaches have been developed to return fresher results for SPARQL queries and at the same time speed up queries [30, 31, 33], but freshness estimates are a major and still current challenge because of the difficulty of predicting the underlying dynamics on the growing Web of Linked Data. Aside from these hybrid approaches, to the best of our knowledge, not much work has been done on predicting the dynamic behavior of SPARQL query results.

Our goal in this work is to then to predict if and/or when the results of a given SPARQL query on a given dynamic Linked Dataset will change, using techniques that are efficient in time and space and offer accurate predictions.

2 STATE OF THE ART

Predicting SPARQL Query Dynamics is related to RDF Data Dynamics, Synchronization and SPARQL Queries. We now discuss important concepts for this work.

2.1 RDF Data Dynamics

Being able to understand and characterize change in RDF data is important for many practical use-cases, such as, Synchronization, Smart Caching, Link Maintenance, Vocabulary Evolution and Versioning, Hybrid Architectures and Vandalism Prevention. When applications locally store descriptions that originate from various remote datasets, awareness of the changes that such descriptions undergo at their origin allows for timely updates, and hence delivering services based on fresh information. Also, an understanding of a dataset's rate of change can inform a decision as to whether or not an application will cache its content locally, or remotely query it on the fly [30, 31].

With respect to the definition of RDF, SPARQL and Linked Data dynamics, there exists a general consensus that it is a function of change and time, and that is where the distinctions begin in terms of the precise level at which to observe the changes (for example, statements [13], entities [29], schema [8], query results [14], etc.) and the discretization of the time or frequency to observe changes (for example, hourly, daily, weekly, etc.). Qualitative approaches attempt to detect changes at different levels of complexity according to their definition of dynamics [25, 28] while quantitative approaches attempt to measure change and dynamicity using mathematical functions [7, 21, 22].

In general, some features found to describe the dynamics of a resource are Pay-Level Domain (PLD) [22, 23], predicates [13, 23], and variants of Characteristic Sets (CS)[4] [11, 22]; these features are correlated with the dynamics of several datasets studied. This knowledge can be used to estimate the freshness of cached data.

2.2 Synchronization

A variety of works have addressed the issue of modeling and consuming dynamic Linked Data from a broad range of perspectives [8, 11–13, 22, 24, 29, 32]. One of the major challenges considered is that of keeping cached copies of remote dynamic data – cached for reasons of efficiency and scalability – up-to-date on the consumer side, which we refer to as the *synchronization problem*.

Some works have addressed the synchronization problem on the publisher side, proposing notification mechanisms that keep registered consumers informed about relevant changes to the data [9, 12, 24]; although such approaches may facilitate strong consistency – meaning that consumers are kept up-to-date with the remote data on the publisher side – they centralize the burden of synchronization on the publisher, potentially leading to scalability issues, particularly for in-demand datasets.

Conversely, a variety of works have looked at building models of remote data that can help to predict which data are most dynamic, and which are most static, indicating which subsets of the data may need to be refreshed from the remote source more often [8, 11, 13, 22, 24, 29, 32]; such works consider changes in RDF datasets at differing levels of granularity, including documents [13], domains [13, 22, 23], predicates [13, 23], characteristic sets [11, 22], etc. Features at different levels of granularity can be fed into different predictive models based on Poisson Processes [29], Markov Chains [34], Empirical Distributions [19], Machine Learning classification and regression [11, 23], as well as a variety of other heuristics [14, 34] and metrics [1, 6, 14].

2.3 Query Analysis

We have seen that RDF data evolve and that we can apply various techniques to learn from past changes and predict certain changes in the future; however, the way we represent RDF data differs from the way we query it. SPARQL is the recommended language to query RDF data, where a query can be made up of multiple algebraic operators that are evaluated on potentially heterogeneous data.

Specifically regarding the dynamics of SPARQL query results, Passant and Mendes [24] proposed sparqlPuSH as a notification framework based on PubSubHubBub (recently standardized as WebSub [9]) aiming for strong consistency. However, the sparqlPuSH implementation presents limitations [15] that make it unapplicable in certain cases and not able to deal with the constantly evolving RDF data available at Web scale.

Rather than aiming for weak consistency, there is an inherent trade-off between centralized approaches that can efficiently answer queries over data cached from parts of the Web [4, 36], and live decentralized approaches that can provide fresher results over the entire Web at the cost of slower response times [10]. Hybrid Approaches try to balance the needs of consistency with the lowest possible latency [5, 30]. These approaches provide responses with a certain trade-off to obtain high consistency like a live integration approach and fast response times through caching.

[4] A characteristic set is the set of predicate terms used to describe a given subject [20].

3 PROPOSED APPROACH

In our literature review we identified two open questions with respect to the state of the art trying to understand and manage the dynamism of Linked Datasets. The questions that we have identified are related to dynamic models and consumption of dynamic datasets but with specific focus on studying the dynamics of SPARQL query results. In this direction, the following research questions are addressed:

(1) How can we identify, model, and understand the dynamics of SPARQL queries and SPARQL query results?
(2) How can we use knowledge about SPARQL query dynamics to increase the consistency of cached query results at a lower cost?

In the following we discuss these high-level research questions in more detail, outlining current work and plans for future work. The framework described in Section 3.1 and the results of Section 4 have been published as a workshop [16] and conference paper [17].

3.1 Framework for Predicting the Dynamics of Query Results

We consider a *dynamic RDF graph* to be a sequence of n discrete versions of an RDF graph denoted $\mathcal{G} \coloneqq (G_1, \ldots, G_n)$; in practice, we assume these versions to have regular intervals (e.g., hourly, daily, weekly, etc.). Further given a SPARQL (1.1) SELECT query Q, we would like to have knowledge of the dynamics of its results. The first such problem we consider – which we call One Shot Change (OSC) – accepts $\mathcal{G}$, Q and a positive integer k as input and outputs a boolean value predicting whether or not the results will change from G_n to some future version G_{n+k} (i.e., $Q(G_n) \neq Q(G_{n+k})$?). The second (more difficult) problem – which we call Time-To-Live (TTL) – accepts only $\mathcal{G}$ and Q and outputs a positive integer k as a prediction for the lowest such value where $Q(G_n) \neq Q(G_{n+k})$. Our problem is thus concerned with predicting if/when the results of a query will change, rather than predicting how – or to what extent – they will change; these latter problems are left for future work.

Architecture: In Figure 1, we provide an overview of a general architecture for making predictions with respect to the OSC prediction task. A SPARQL query Q and a dynamic RDF graph $\mathcal{G}$ are given as input. The system then extracts a feature vector $(f_1, \ldots, f_k)$. from these inputs and feeds them into a pre-trained binary classifier to make the OSC prediction. The query features $(f_1, \ldots, f_i)$ are extracted online from the query itself. The predicate and degree-of-change features $(f_{i+1}, \ldots, f_j)$ are extracted from a statistical description $\mathrm{d}(\mathcal{G})$ of $\mathcal{G}$, whose details will be described later; in practice, $\mathrm{d}(\mathcal{G})$ can be computed and maintained offline in an incremental manner, requiring only the two most recent versions of $\mathcal{G}$ to be updated. Finally, the results features $(f_{j+1}, \ldots, f_k)$ require as input the full historical results of Q for each version of $\mathcal{G}$ (which we denote by $Q(\mathcal{G})$); this must be computed online. The binary classifier is pre-trained over a given set of queries $\boldsymbol{Q}$ for which ground truths are computed over withheld versions.

*Query features (*Q*).* include statistics about the query. Formally, given a query Q, we define that $\mathrm{Q}(Q) = (n_T, n_V, n'_V, \vec{o})$, where n_T denotes the number of triple patterns in Q, n_V denotes the number of variables in Q, n'_V denotes the number of projected variables in Q, and $\vec{o}$ is a one-hot encoded vector that denotes the SPARQL operators (OPTIONAL, UNION, etc.) used by Q. We hypothesise that such features may be useful for predicting dynamics.

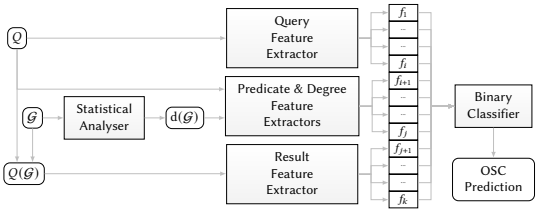

Figure 1: Proposed architecture for predicting change OSC given a query Q and dynamic RDF graph $\mathcal{G}$

*Predicate features (*P*):* include statistics about how frequently and how many triples matching the predicates used in the query change. More formally, let $G_1 \oplus G_2 = G_1 \setminus G_2 \cup G_2 \setminus G_1$ denote the symmetric difference of the graphs (the set of triples which are in either of the graphs G_1 and G_2, but not in their intersection). Further let $\sigma_{\mathrm{P}=p}(G) = \{(x, y, z) \in G \mid p = y\}$ denote the triples in G using the predicate p. Now, given a dynamic RDF graph $\mathcal{G} = (G_1, \ldots, G_k)$, we denote by $\Delta(\mathcal{G}, p) = \sum_{i=1}^{k-1} \frac{|\sigma_{\mathrm{P}=p}(G_i \oplus G_{i+1})|}{|\sigma_{\mathrm{P}=p}(G_i \cup G_{i+1})|}$ the sum of the ratios of triples for p that changed between each consecutive pair of versions, such that the higher the value for $\Delta(\mathcal{G}, p)$, the more dynamic the triples associated with the predicate. Next let $\mathrm{preds}(Q)$ denote the set of IRIs used as predicates in Q. Now we define $\mathrm{P}(\mathcal{G}, Q) = \frac{\sum_{p \in \mathrm{preds}(Q)} \Delta(\mathcal{G}, p)}{|\mathrm{preds}(Q)|}$, taking the mean value of $\Delta(\mathcal{G}, p)$ for all predicates in Q. Predicate features only require lightweight statistics about the dynamic RDF graph since the number of unique predicates – even in large RDF graphs – tends to be relatively small.

*Degree-of-change features (*D*):* include statistics about the variability in the number of results returned by the query across the historical versions. Specifically, given a query Q and an RDF graph G, we denote by $\mathrm{card}(\cdot, \cdot)$ a *cardinality estimation function*, where $\mathrm{card}(Q, G)$ estimates the (bag) cardinality of $|Q(G)|$. Any such function can be used; currently we use a statistic method similar to Postgres[5] that only requires a statistic description $d(\mathcal{G})$ and not $\mathcal{G}$ itself. We use the function $\mathrm{card}(Q, G)$ to estimate the dynamics of a query by comparing the number of results for the query that depend on data that did not change between two versions, versus the number of results generated over the union of the two versions. Specifically, given a dynamic RDF graph $\mathcal{G} = (G_1, \ldots, G_k)$, we sum the ratios $\mathrm{D}(\mathcal{G}, Q) = \sum_{i=1}^{k-1} 1 - \frac{\mathrm{card}(Q, G_i \cap G_{i+1})}{\mathrm{card}(Q, G_i \cup G_{i+1})}$ to estimate the degree-of-change, i.e., the ratio of query results that are sensitive to changes between versions.

*Results features (*R*):* include statistics about the historical results for the query. These features are conceptually the simplest, where we simply count the number of times the results for the query Q changed between the pairs of consecutive versions of the dynamic RDF graph. Formally, given a query Q and a dynamic RDF graph with k known versions $\mathcal{G} = (G_1, \ldots, G_k)$ we define this feature as:

[5] https://www.postgresql.org/docs/current/static/planner-stats-details.html

$R(\mathcal{G}, Q) = |\{i \in \{1, \ldots, k-1\} : Q(G_i) \neq Q(G_{i+1})\}|$. Though conceptually the simplest, we expect that this feature will also provide the most useful information for OSC prediction. Conversely, it incurs the highest overhead for an unseen query Q, requiring maintaining all data for $G_1, \ldots, G_k$ offline and evaluating the queries used for training over these versions, as well as evaluating $Q(G_1), \ldots, Q(G_k)$ online (when the query is received).

Hypothesis: We hypothesise that the methods we propose follow a trade-off for predictions, where more accurate predictions imply greater overheads. On one side, we have query features, which require no knowledge of the data, and thus involve the least overhead, but provide less accurate predictions as a result. Next we have features about the dynamics of predicates used in the query, which require high-level statistics about how the triples involving a particular predicate change. Thereafter, the degree-of-change measures require more detailed statistics, but allow for computing more detailed meta-data regarding a particular query's sensitivity to changes in the graph. Finally, we have knowledge of historical results, which for an unseen query, requires evaluating the query on several historical versions, and maintaining indexes over those versions, thus implying a much higher overhead, but with the benefit of having much more detailed information about how its results have changed over past versions.

3.2 Smart Caching Framework

With a framework for modelling and predicting SPARQL query dynamics in place, we believe that we increase the consistency of cached SPARQL query results at a lower cost by using this framework. In that sense, we propose to develop a configurable Smart Caching Framework that contains best approaches that result from our aforementioned research.

4 RESULTS

We have implemented the architecture and features described in Section 3.1, for which we now present evaluation. We focus on the accuracy of the features for OSC prediction.

This evaluation aims to ascertain the quality of predictions as to whether or not the results for a query will change in the next version considering these features. We begin by describing the datasets and queries used [18].

Datasets. We first consider a dataset of 17 weekly versions of Wikidata, spanning from 2019/11/14 (containing 4.4 billion triples) to 2020/03/05 (containing 4.8 billion triples). The number of triples grew by 9.2% during the time period, where approximately 4 times more triples are added, on average, than removed. A total of 78 billion triples are present in all versions. We use 141 queries that were sourced from the user-contributed example queries published by the Wikidata query service. There were 1300 changes in results between pairs of versions (out of a possible 2256 comparisons). The results for 20 queries never change, while 56 change in all versions.

We consider a second dataset based on DBpedia Live, where we use the changesets to build 18 daily versions, spanning from 2019-07-01 (containing 593.6 million triples), until 2019-07-18 (containing 590.3 million triples). In this case we see more deletions than insertions, and fewer changes overall, when compared with Wikidata. Overall, the DBpedia versions contain 10.7 billion triples. We use a set of 254 queries from the ones extracted by Knuth et al. [14] from the LSQ dataset [26], composed of queries evaluated by the DBpedia SPARQL endpoint. Of these 254 queries, there were 416 changes in results between pairs of versions (out of a possible 4,318 comparisons). Of the 254 queries, 54 have at least one change: 16 queries change each time while 21 change only once.

Table 1: F_1-score for OSC on the Wikidata and DBpedia datasets considering a window size of three.

	Classifier	Q	P	D	R	QPD	QPDR
Wikidata	*Random Baseline*	0.509	0.499	0.497	0.482	0.496	0.5
	Decision Trees	0.549	**0.554**	0.66	**0.855**	**0.61**	0.709
	Naive Bayes	0.522	0.36	0.478	0.825	0.498	0.67
	Nearest Neighbours	0.547	0.532	**0.691**	0.837	0.537	**0.83**
	Linear SVM	**0.582**	0.441	0.387	0.825	0.603	0.827
DBpedia	*Random Baseline*	0.503	0.514	0.494	0.498	0.486	0.491
	Decision Trees	**0.57**	0.488	**0.567**	**0.928**	**0.535**	0.888
	Naive Bayes	0.475	**0.532**	0.473	0.902	0.489	0.809
	Nearest Neighbours	0.519	0.474	0.507	**0.928**	0.481	0.904
	Linear SVM	0.474	0.474	0.504	**0.928**	0.509	**0.925**

Binary classification models. We experiment with well-known machine learning classifiers. We also include a Random Baseline for comparison. We train and compare classifiers for different window sizes. For each query we evaluate whether or not the change in the query results can be predicted from the set of features extracted from the query itself and from the preceding dynamic graph. We split the data by query into 80% for training and 20% for tests, using 5-fold cross-validation to avoid overfitting.

Results. Table 1 presents the results for OSC prediction on the Wikidata and DBpedia datasets. For reasons of space, we include only F_1 scores and windows of size three. We compare the results for six feature sets: query (Q), property (P), degree-of-change (D), historical results (R), all features without historical results (QPD) and all features (QPDR). We highlight the best results in bold.

Comparing different feature sets, we see that statistics based on historical query results (R) are the most important, and enable (by far) the most accurate predictions. In fact, the predictions with only results features are considerably better than the predictions combining all other features. Conversely, we find that query features and predicate features provide only slightly better predictions versus the random baseline. In the case of Wikidata, degree-of-change features offer notably better predictions than query or predicate-based features, ethe quality specially when using the Nearest Neighbours or Decision Trees classifier. However, of predictions drops for DBpedia, which we believe to be due to the relative sparsity of changes in the query results of the dataset, and also the more non-monotonic nature of changes vs. Wikidata.

5 METHODOLOGY

Our next steps are aimed at improving our estimations of when the results of a SPARQL query will change on a dynamic RDF dataset. These improvements involve support for additional SPARQL operators, refinements of the cardinality estimations, methods for predicting TTL, as well as probabilistic aggregations. We also want to evaluate in more detail the time/space requirements of the different features, as well as performing tests on other datasets. Then we

want to evaluate the feasibility of our prediction schemes in a real-world query caching system. More specifically, the steps currently underway are:

(1) *Develop methods for estimating the dynamics of query results based on data dynamics, but without using full historical data.*
(2) Implement a dynamics based query caching system.

This methodology corresponds to both: work already tackled or in progress (stressed in italics) and future plans for work yet to begin. Evaluation will be based on real-world data and queries, using historical versions and query logs from Wikidata and DBpedia.

6 CONCLUSIONS AND FUTURE WORK

This Ph.D. proposal is motivated by the goal of improve the way applications access dynamic Linked Data, making queries more efficient while ensuring fresher results. Our first aim has been to understand how to model the dynamics of SPARQL query results, developing a framework to predict if (and eventually when) the results of a query will change due to changes in the underlying data. There exists a trade-off between the overhead of the framework and the accuracy of prediction: extracting richer data provides better features for prediction, but at the cost of extracting them.

Our results confirm that features based on historical results provide by far the most accurate predictions. However, such features incur considerable overhead that may be unjustifiable in use-cases such as caching. Up until now, using the degree-of-change feature allow estimates of the next best alternative in terms of accuracy; such features require a statistical summary rather than a complete index of historical versions.

Our immediate next steps is to explore other possible features that may help to improve predictions without needing historical results. Thereafter, we hope to extend our techniques to cover not only OSC predictions, but also TTL predictions. Our final aim is to develop a general Smart Caching Framework that integrates lower cost features to improve the consistency of cached data and thus the efficiency of graph query systems. We are also interested to explore other possible use-cases for our framework.

Acknowledgements. This work was supported by CONICYT-PCHA /Doctorado Nacional/2017-21171070 and by the Millennium Institute for Foundational Research on Data (IMFD).

REFERENCES

[1] Usman Akhtar, Muhammad Bilal Amin, and Sungyoung Lee. 2017. Evaluating scheduling strategies in LOD based application. In *APNOMS, Korea (South)*. IEEE.
[2] Carlos Buil Aranda, Aidan Hogan, Jürgen Umbrich, and Pierre-Yves Vandenbussche. 2013. SPARQL Web-Querying Infrastructure: Ready for Action?. In *The Semantic Web - ISWC 2013*. Springer, 277–293.
[3] Sören Auer, Jan Demter, Michael Martin, and Jens Lehmann. 2012. LODStats - An Extensible Framework for High-Performance Dataset Analytics. In *EKAW 2012, Ireland, October 8-12, 2012. Proceedings*. Springer, 353–362.
[4] Roger Castillo, Christian Rothe, and Ulf Leser. [n.d.]. *RDFMatView: Idexing RDF Data for SPARQL Queries*.
[5] Soheila Dehghanzadeh, Josiane Xavier Parreira, Marcel Karnstedt, Jürgen Umbrich, Manfred Hauswirth, and Stefan Decker. 2014. Optimizing SPARQL Query Processing on Dynamic and Static Data Based on Query Time/Freshness Requirements Using Materialization. In *JIST 2014, Thailand*. Springer, 257–270.
[6] Renata Queiroz Dividino, Thomas Gottron, and Ansgar Scherp. 2015. Strategies for Efficiently Keeping Local Linked Open Data Caches Up-To-Date. In *ISWC 2015, USA, October 11-15, 2015*. Springer, 356–373.
[7] Renata Queiroz Dividino, Thomas Gottron, Ansgar Scherp, and Gerd Gröner. 2014. From Changes to Dynamics: Dynamics Analysis of Linked Open Data Sources. In *PROFILES@ESWC 2014, Anissaras, Crete, Greece, May 26, 2014*.
[8] Renata Queiroz Dividino, Ansgar Scherp, Gerd Gröner, and Thomas Grotton. 2013. Change-a-LOD: Does the Schema on the Linked Data Cloud Change or Not?. In *COLD 2013, Sydney, Australia, October 22, 2013*. CEUR-WS.org.
[9] Julien Genestoux, Brad Fitzpatrick, Brett Slatkin, and Martin Atkins. 2018. WebSub. W3C Recommendation. https://www.w3.org/TR/websub/.
[10] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu. 2011. View Selection in Semantic Web Databases. *PVLDB* 5, 2 (2011), 97–108.
[11] Larry González and Aidan Hogan. 2018. Modelling Dynamics in Semantic Web Knowledge Graphs with Formal Concept Analysis. In *WWW 2018, France*. ACM.
[12] Luis Daniel Ibáñez, Hala Skaf-Molli, Pascal Molli, and Olivier Corby. 2014. ColGraph: Towards Writable and Scalable Linked Open Data. In *ISWC 2014, Italy*.
[13] Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan. 2013. Observing Linked Data Dynamics. In *ESWC 2013, France*. Springer.
[14] Magnus Knuth, Olaf Hartig, and Harald Sack. 2016. Scheduling Refresh Queries for Keeping Results from a SPARQL Endpoint Up-to-Date (Short Paper). In *OTM - CoopIS, C&TC, and ODBASE, Greece*. Springer, 780–791.
[15] Magnus Knuth, Dinesh Reddy, Anastasia Dimou, Sahar Vahdati, and George Kastrinakis. 2015. Towards Linked Data Update Notifications Reviewing and Generalizing the SparqlPuSH Approach. In *NoISE, (ESWC)Slovenia*. CEUR-WS.org.
[16] Alberto Moya Loustaunau and Aidan Hogan. 2019. Estimating the Dynamics of SPARQL Query Results Using Binary Classification. In *QuWeDa, (ISWC)*. 5–20.
[17] Alberto Moya Loustaunau and Aidan Hogan. 2021. Predicting SPARQL Query Dynamics. In *K-CAP '21, USA*. ACM, 161–168.
[18] Alberto Moya Loustaunau and Aidan Hogan. 2021. Online material. GitHub Page. https://amoya87.github.io/sparqldynamics/.
[19] Sebastian Neumaier and Jürgen Umbrich. 2016. Measures for Assessing the Data Freshness in Open Data Portals. In *OBD, Vienna, Austria*. IEEE Computer Society.
[20] Thomas Neumann and Guido Moerkotte. 2011. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *ICDE, Germany*.
[21] Chifumi Nishioka and Ansgar Scherp. 2015. Temporal Patterns and Periodicity of Entity Dynamics in the Linked Open Data Cloud. In *K-CAP, USA*. ACM.
[22] Chifumi Nishioka and Ansgar Scherp. 2016. Information-theoretic Analysis of Entity Dynamics on the Linked Open Data Cloud. In *PROFILES, ESWC, Greece*.
[23] Chifumi Nishioka and Ansgar Scherp. 2017. Keeping Linked Open Data caches up-to-date by predicting the life-time of RDF triples. In *Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, August 23-26, 2017*. ACM, 73–80.
[24] Alexandre Passant and Pablo N. Mendes. 2010. sparqlPuSH: Proactive Notification of Data Updates in RDF Stores Using PubSubHubbub. In *Proceedings of the Sixth Workshop on Scripting and Development for the Semantic Web, Greece*.
[25] Yannis Roussakis, Ioannis Chrysakis, Kostas Stefanidis, Giorgos Flouris, and Yannis Stavrakas. 2015. A Flexible Framework for Understanding the Dynamics of Evolving RDF Datasets. In *ISWC, USA*. Springer, 495–512.
[26] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2015. LSQ: The Linked SPARQL Queries Dataset. In *International Semantic Web Conference (ISWC)*. Springer, 261–269.
[27] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the Linked Data Best Practices in Different Topical Domains. In *ISWC, Italy*.
[28] Kostas Stefanidis, Giorgos Flouris, Ioannis Chrysakis, and Yannis Roussakis. 2016. D2V - Understanding the Dynamics of Evolving Data: A Case Study in the Life Sciences. *ERCIM News* 2016, 105 (2016).
[29] Jürgen Umbrich, Michael Hausenblas, Aidan Hogan, Axel Polleres, and Stefan Decker. 2010. Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources. In *LDOW, WWW2010, USA*. CEUR-WS.org.
[30] Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira. [n.d.]. Hybrid SPARQL Queries: Fresh vs. Fast Results. In *ISWC 2012, USA*.
[31] Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira. 2012. Freshening up while Staying Fast: Towards Hybrid SPARQL Queries. In *EKAW, Ireland Proceedings*. Springer, 164–174.
[32] Jürgen Umbrich, Marcel Karnstedt, and Sebastian Land. 2010. Towards Understanding the Changing Web: Mining the Dynamics of Linked-Data Sources and Entities. In *LWA 2010 - Lernen, Wissen & Adaptivität*. 159–162.
[33] Jürgen Umbrich, Marcel Karnstedt, Josiane Xavier Parreira, Axel Polleres, and Manfred Hauswirth. 2012. Linked Data and Live Querying for Enabling Support Platforms for Web Dataspaces. In *ICDE, USA*. IEEE Computer Society, 23–28.
[34] Jürgen Umbrich, Nina Mrzelj, and Axel Polleres. 2015. Towards capturing and preserving changes on the Web of Data. In *DIACHRON Workshop on Managing the Evolution and Preservation of the Data Web, ESWC 2015, Slovenia*. 50–65.
[35] Pierre-Yves Vandenbussche, Jürgen Umbrich, Luca Matteis, Aidan Hogan, and Carlos Buil Aranda. 2017. SPARQLES: Monitoring public SPARQL endpoints. *Semantic Web* 8, 6 (2017), 1049–1065.
[36] Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. 2016. Triple Pattern Fragments: A low-cost knowledge graph interface for the Web. *J. Web Sem.* 37-38 (2016), 184–206.
[37] Wei Emma Zhang, Quan Z. Sheng, Lina Yao, Kerry Taylor, Ali Shemshadi, and Yongrui Qin. 2018. A Learning-Based Framework for Improving Querying on Web Interfaces of Curated Knowledge Bases. *ACM Trans. Internet Techn.* (2018).