Interest-based RDF Update Propagation

Kemele M. Endris, Sidra Faisal, Fabrizio Orlandi, Sören Auer, Simon Scerri

University of Bonn & Fraunhofer IAIS, Bonn, Germany
{endris,faisals,orlandi,auer,scerri}@cs.uni-bonn.de

Abstract. Many LOD datasets, such as DBpedia and LinkedGeoData, are voluminous and process large amounts of requests from diverse applications. Many data products and services rely on full or partial local LOD replications to ensure faster querying and processing. Given the evolving nature of the original and authoritative datasets, to ensure consistent and up-to-date replicas frequent replacements are required at a great cost. In this paper, we introduce an approach for interest-based RDF update propagation, which propagates only interesting parts of updates from the source to the target dataset. Effectively, this enables remote applications to 'subscribe' to relevant datasets and consistently reflect the necessary changes locally without the need to frequently replace the entire dataset (or a relevant subset). Our approach is based on a formal definition for graph-pattern-based interest expressions that is used to filter interesting parts of updates from the source. We implement the approach in the iRap framework and perform a comprehensive evaluation based on DBpedia Live updates, to confirm the validity and value of our approach.

Keywords: Change Propagation, Dataset Dynamics, Linked Data, Replication

1 Introduction

In recent years, there has been an increasing number of structured data published on the Web as Linked Open Data (LOD). As of 2014, the size of the LOD cloud consisted of more than 1.000 published datasets comprising almost 100 Billion triples[1]. Many of these datasets are huge and process large amount of requests from diverse applications. Providing services on top of these datasets is becoming a challenge due to the lack of service levels regarding the availability of datasets [11] and restrictions imposed by the publisher on the type of query forms and number of results[2]. Replication of Linked Data datasets enhances flexibility of information sharing and integration infrastructures. Since hosting a replica of large datasets is costly, organizations might want to host only a relevant subset of the data, for example, using approaches such as *RDFSlice* [4]. However, due to the evolving nature of these datasets, maintaining a consistent and up-to-date

[1] http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/
[2] https://lists.w3.org/Archives/Public/public-lod/2011Aug/0028.html

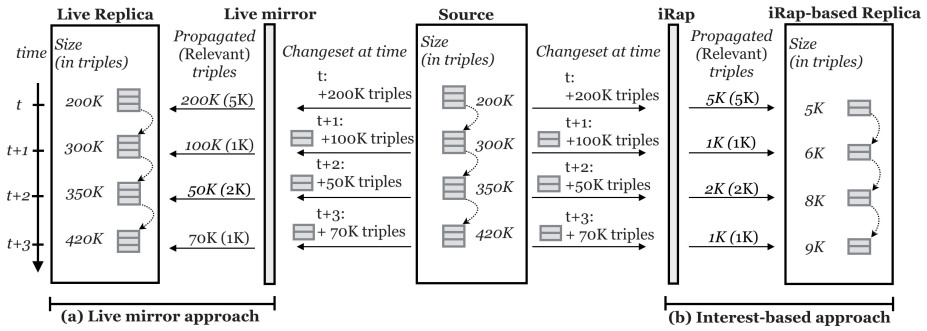


Fig. 1: Changeset propagation approaches: left part (a) – Live mirror approach; right part (b)– Interest-based approach

replica of the relevant data is a major challenge. Resources in a dataset might be added, updated, or removed. Applications consuming these datasets should be capable of dealing with such updates to keep their local copies consistent.

Typically, dataset mirror applications propagate updates published by the source dataset to a target dataset (replica). For instance, the *DBpedia Live mirror tool*[3] propagates all changes to a target dataset, so that at any point of time the target dataset contains the same triples as the DBpedia Live dataset. However, for example, an application interested in athletes uses only 268,773 out of 4,584,616 instances of the English DBpedia 2014 dataset[4]. In this paper, we present an approach for interest-based update propagation, which is based on the specification of data interests by a target application. Based on such interest expressions all updates are evaluated and only those changes satisfying target applications' interest are shipped to the target dataset. An interest-based update propagation could significantly reduce the amount of data to be shipped and managed at the application side and thus lower the barrier for the deployment of Linked Data applications. We provide a thorough formalization of our approach.

Figure 1 shows the propagation of unfiltered data from a source to a target, referred to as *Live Replica* (part *(a)* on the left). This approach propagates all the updates irrespective of the relevance or usefulness of the data. Whereas, using iRap (interest-based RDF update propagation) framework the source-to-target data propagation (*iRap-based Replica* in Figure 1 part *(b)* on the right) is filtered. With this interest-based approach only relevant data is being transfered. Our evaluation shows, that the data required to be transfered and handled by applications can be reduced by several orders of magnitude thus substantially lowering the Linked Data re-use barrier.

The article is structured as follows: section 2 extensively describes the formalization for our framework. section 3 and section 4 discusses the implementation and evaluation of the iRap framework in detail. section 5 describes the related work. Finally, section 6 concludes and proposes directions for future work.

[3] `https://github.com/dbpedia/dbpedia-live-mirror`

[4] `http://wiki.dbpedia.org/services-resources/datasets/dataset-statistics`

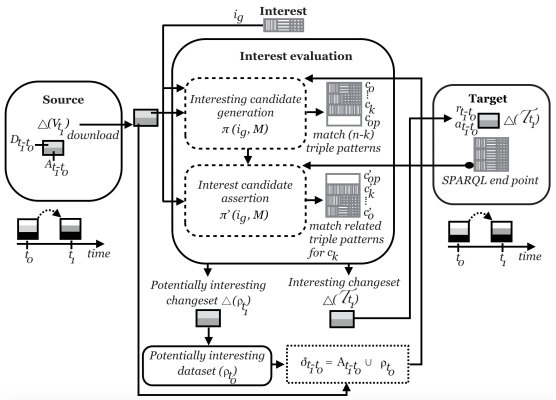


Fig. 2: Formalization overview of the interest-based RDF update propagation.

2 Formalization of Interest-based RDF Updates

Figure 2 illustrates the overall interest-based RDF Update Propagation approach; summarizing the concepts defined through the formalization. Interest evaluation takes place over the input set of deleted ($D_{t_1-t_0}$) and added ($At_1 - t_0$) triples from the source dataset (V_{t_1}) in between time interval (t_0, t_1). Since updates do not only contain interesting and uninteresting parts, but also triples which can become interesting along with subsequent updates. We have to compute and store these sets of *potentially interesting triples* (see Definition 8) and take them in subsequent update assessments into account.

For our formalization we will use the standard notations **I**, **B**, **L** and *Var* for the disjoint sets of all IRIs, blank nodes, literals (typed and untyped) and variables respectively. An *RDF graph* V is a finite set of RDF triples, i.e, $V \subset (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$. In this paper we use the terms RDF graph, *RDF dataset*, and *dataset* interchangeably. An *evolving dataset* V^g is a dataset identified using the persistent IRI g whose content changes over time. V_t^g denotes a specific revision of V^g at a particular time t. For simplicity, we will just refer to V_t instead of V_t^g.

Definition 1 (BGP). *A SPARQL basic graph pattern (BGP) expression is defined recursively as follows:*

1. *a triple pattern* $tp \in (\boldsymbol{I} \cup \boldsymbol{B} \cup Var) \times (\boldsymbol{I} \cup Var) \times (\boldsymbol{I} \cup \boldsymbol{B} \cup \boldsymbol{L} \cup Var)$ *is a BGP*
2. *the expression (P1 AND P2) is a BGP, where P1 and P2 are themselves BGPs*
3. *the expression (P FILTER E) is a BGP, where P is a BGP and E is a SPARQL filter expression that evaluates to boolean value.*

Definition 2 (Non-disjoint BGP). *A non-disjoint BGP is a BGP that represents a connected graph.*

An optional graph pattern (OGP) is syntactically specified with the OPTIONAL keyword applied to a graph pattern. A set of triple patterns in a BGP must

```
dbr:Marcel              dbp:goals     1 .
dbr:Marcel              dbo:team      dbr:FNFT .
dbr:Tim%02              foaf:name
                          "Tim Berners-Lee" .
dbr:Cristiano_Ronaldo   dbo:goals     96 .
```

Listing (1.1) File 000001.removed.nt

```
dbr:Cristiano_Ronaldo  dbo:goals 216 .
dbr:Barack_Obama       foaf:name "Barack Obama" .
dbr:Barack_Obama       foaf:homepage
                      "http://www.barackobama.com/" .
dbr:Rio_Ferdinand      a         foaf:Person .
dbr:Rio_Ferdinand      a         dbo:Athlete .
dbr:Rio_Ferdinand      dbp:goals 2 .
dbr:Arvid_Smit         a         dbo:Athlete .
```

Listing (1.2) File 000001.added.nt

Fig. 3: Changeset files published by DBpedia Live extractor

match for there to be a solution whereas triple patterns in OGP may extend the solution but their non-binding nature means that they cannot reject it [1].

Definition 3 (Partial Matches). *Partial matches are a set of triples that does not fully match the BGP but matches at least one triple pattern in BGP or OGP of a query.*

Triples added to, and removed from, an evolving dataset within a time-frame are called *changeset* for a dataset within that time-frame.

Definition 4 (Changeset). *Let V_{t_1} be an evolving dataset at time t_1. A changeset $\Delta(V_{t_1})$, between V_{t_0} and V_{t_1}, where $t_0 < t_1$, is defined as:*

$$\Delta(V_{t_1}) = \langle D_{t_1-t_0}, A_{t_1-t_0} \rangle$$

where: $D_{t_1-t_0}$ is a set of removed triples from V_{t_0} between time-points t_0 and t_1, and $A_{t_1-t_0}$ is a set of added triples to V_{t_0} between time-points t_0 and t_1.

Changesets can be computed using the difference between two versions of the RDF dataset. The result of this computation gives the removed triples, $D_{t_1-t_0} = V_0 \backslash V_1$, and added triples, $A_{t_1-t_0} = V_1 \backslash V_0$, between given dataset revisions V_{t_0} and V_{t_1}. Datasets can be accompanied with a tool that publishes changesets at real-time, so that users can download these changesets and synchronize their local replicas. For instance, DBpedia publishes updates in a public changesets folder (`http://live.dbpedia.org/changesets/`).

Example 1. Let us assume two files (Listing 1.1 and Listing 1.2) are being published by the DBpedia Live extractor for the changes made on Feb 06, 2015 between 05:00 PM (t_0) and 05:02 PM (t_1). A changeset $\Delta(V_{t_1})$ for the DBpedia Live dataset between t_0 and t_1, contains $D_{05:02-05:00} = 000001.removed.nt$ and $A_{05:02-05:00} = 000001.added.nt$.
That is, $\Delta(V_{05:02}) = \langle 000001.removed.nt,\ 000001.added.nt \rangle$.

Definition 5 (Changeset Propagation). *A changeset propagation is a function υ that transforms a given dataset V_{t_0} to a new dataset V_{t_1} by applying a changeset, $\Delta(V_{t_1})$. That is: $\upsilon(V_{t_0}, \Delta(V_{t_1})) = (V_{t_0} \backslash D_{t_1-t_0}) \cup A_{t_1-t_0} = V_{t_1}$*

The changeset propagation function υ, for example, deletes the triples in 000001.removed.nt from the target dataset and then inserts all triples from 000001.added.nt.

This order of operation (deleted first) ensures that inserted triples are not removed again immediately. If an organization maintaining a replica wants to host only a subset of the original dataset, it needs to obtain only relevant updates for this subset. For that purpose, we specify *interests* to subscribe to 'interesting' changes only. During interest registration, an organization provides information about the source dataset to synchronize with, a target dataset endpoint that supports SPARQL Update to propagate interesting changes, and an interest query to select relevant parts of a changeset.

Definition 6 (Interest Expression). *An interest expression over an evolving dataset, V_t, is defined as: $i_g = \langle \tau, b, op \rangle$ where g is an IRI identifying an evolving RDF dataset V_t, τ is an IRI identifying the target dataset endpoint, b is a non-disjoint BGP, and op is an optional graph pattern (OGP) connected to b.*

Example 2. An interest expression for a list of an athlete with information about goals scored, and optionally their homepage, is expressed as follows:

- g = *http://live.dbpedia.org/changesets*
- τ = *http://localhost:3030/target/sparql*
- b = `{ ?a a dbo:Athlete . ?a dbp:goals ?goals . }`
- op = `{ ?a foaf:homepage ?page . }`

The equivalent interest expression SPARQL query will be:

```
SELECT * WHERE { ?a a dbo:Athlete . ?a dbp:goals ?goals . OPTIONAL { ?a foaf:homepage ?page . } }
```

In order to initialize a local data store, i.e., the target dataset, SPARQL CONSTRUCT queries can be used by employing the interest expression's BGPs to extract and load a subset of the source dataset. Then interest expressions are registered with our iRap framework to retrieve interesting updates from the source dataset. iRap evaluates interest expressions over changesets being published along with the source dataset. Without a restriction of generality, we assume interest expressions here to be static for the lifetime of a target dataset, since an evolution of interest expressions can be simulated by removal and addition. The result of executing an interest evaluation for an interest expression against a changeset are three sets or triples: *1. interesting, 2. potentially interesting, and 3. uninteresting* triples.

Definition 7 (Interesting Triples). *Interesting triples are all triples comprised in full matches of the BGP and possibly OGP of an interest expression, i_g, against the sets of added or deleted triples of a changeset. Interesting triples originating from the first element (i.e., removed triples ($D_{t_1-t_0}$)) of a changeset, $\Delta(V_{t_1})$, are called interesting-removed triples. Interesting triples originating from the second element (i.e., added triples ($A_{t_1-t_0}$)) of a changeset, $\Delta(V_{t_1})$, are called interesting-added triples.*

In addition to parts of an changeset for which the 'interestingness' can be immediately decided, there might also be parts, which are *potentially interesting* since, i) the missing parts to render them as interesting are already contained in the target knowledge base or ii) they will be propagated in subsequent updates.

Definition 8 (Potentially Interesting Triples). *Potentially interesting triples are triples comprised in partial matches of the BGP or in OGP of interest expression, i_g:*

- *Potentially interesting triples originating from the first element (i.e., removed triples ($D_{t_1-t_0}$)) of a changeset $\Delta(V_{t_1})$, are called potentially interesting-removed triples.*
- *Potentially interesting triples originating from the second element (i.e., added triples ($A_{t_1-t_0}$)) of a changeset, $\Delta(V_{t_1})$, are called potentially interesting-added triples.*

Potentially interesting triples can become interesting if triples missing in the changeset, but required for a full BGP match, are found in the target dataset or in subsequent changesets. Finally, there are triples in the changeset that are neither interesting nor potentially interesting.

Definition 9 (Uninteresting Triples). *Uninteresting triples are triples that do not match any triple pattern in a BGP or OGP of any interest expression, i_g, against the sets of added or deleted triples of a changeset.*

Uninteresting triples are not interesting at the moment and can never become interesting with subsequent changesets. iRap uses an interest query to select candidate triples from a changeset and to assert from a target dataset. These candidates are retrieved in decreasing order of number of matching BGP triple patterns of interest expressions and triples that match any part of optional graph patterns.

Definition 10 (Interest Candidate Generation). *An interest candidate generation is the extraction of matching triples from a changeset for a non-disjoint combination of triple patterns in BGP of an interest expression, i_g. The result of this extraction is an $(n+1)$-tuple with decreasing order of matching:*

$$\pi(i_g, M) = \langle c_0, c_1, ..., c_{n-1}, c_{op} \rangle$$

where:

- *M is a set of removed (respectively added) triples in a changeset,*
- *n is the number of triple patterns in the BGP of interest expression, i_g,*
- *c_k is a set of candidate triples in M that match $n-k$ $(0 \leqslant k < n)$ triple patterns of the BGP of the interest expression, i_g, and*
- *c_{op} is a set of candidate triples in M that match at least one triple pattern in the OGP of interest expression, i_g, but none of the triple patterns in the BGP.*

Example 3. An interest candidate generation for the interest expression i_g from Example 2 over the changeset from Example 1 gives the following result:

1. $\pi(i_g, D_{05:02-05:00}) = \langle c_0, c_1, c_{op} \rangle$ where:
   $c_0 = \varnothing$
   c_1 = `dbr:Marcel dbp:goals 1. dbr:Cristiano_Ronaldo dbo:goals 96.`
   $c_{op} = \varnothing$

2. $\pi(i_g, A_{05:02-05:00}) = \langle c_0, c_1, c_{op} \rangle$ where:
   c_0 = `dbr:Rio_Ferdinand a dbo:Athlete . dbr:Rio_Ferdinand dbp:goals 10.`
   c_1 = `dbr:Cristiano_Ronaldo dbp:goals 216 . dbr:Arvid_Smit a dbo:Athlete.`
   c_{op} = `dbr:Barack_Obama foaf:homepage "http://www.barackobama.com".`

Now an interest candidate assertion verifies candidate triples with respect to all triple patterns in the BGP of an interest expression.

Definition 11 (Interest Candidate Assertion). *The candidate assertion function extracts missing triples for the candidate, c_i of $\pi(i_g, M)$ of an interest expression i_g from the target dataset, τ_{t_0}:*

$$\pi'(i_g, M) = \left\langle c'_{op}, c'_{n-1}, ..., c'_1, c'_0 \right\rangle$$

where:

- *M is a set of removed (respectively added) triples in a changeset,*
- *n is the number of triple patterns in the BGP of interest expression, i_g,*
- *c'_{op} is a set of triples from target dataset, τ, that matches the missing optional graph patterns for candidate c_0, of $\pi(i_g, M)$,*
- *c'_k is a set of triples from target dataset, τ, that matches the missing triple patterns for candidate c_{n-k}, where $0 < k < n$, of $\pi(i_g, M)$, and*
- *c'_0 is a set of triples from target dataset, τ, that matches all triple patterns in BGP of interest expression for candidate c_{op}, of $\pi(i_g, M)$.*

Example 4. Let the target dataset, τ_{t_0}, at time t_0 contains the following triples:

```
#Target dataset at time t0 = 05:00 PM Feb 06, 2015
dbr:Marcel              a              dbo:Athlete .
dbr:Marcel              dbp:goals      1 .
dbr:Cristiano_Ronaldo   a              dbo:Athlete .
dbr:Cristiano_Ronaldo   dbo:goals      96 .
dbr:Cristiano_Ronaldo   foaf:homepage  "http://cristianoronaldo.com" .
```

An interest candidate assertion for interest candidates generated in Example 3 yields the following result:

1. $\pi'(i_g, D_{05:02-05:00}) = \left\langle c'_{op}, c'_1, c'_0 \right\rangle$ where:
   $c'_{op} = \varnothing$
   c'_1 = `dbr:Marcel a dbo:Athlete .`
   `dbr:Cristiano_Ronaldo a dbo:Athlete .`
   `dbr:Cristiano_Ronaldo foaf:homepage "http://cristianoronaldo.com" .`
   $c'_0 = \varnothing$
2. $\pi'(i_g, A_{05:02-05:00}) = \left\langle c'_{op}, c'_1, c'_0 \right\rangle$ where:
   $c'_{op} = \varnothing$
   c'_1 = `dbr:Cristiano_Ronaldo a dbo:Athlete .`
   `dbr:Cristiano_Ronaldo foaf:homepage "http://cristianoronaldo.com" .`
   $c'_0 = \varnothing$

The interest evaluation over a changeset $\Delta(V_{t_1})$ is performed in two steps. First, interest expressions are evaluated against removed triples of a changeset as $d(i_g, D_{t_1-t_0})$, see Definition 12. Second, interest expressions are evaluated

against added triples of a changeset as $\alpha(i_g, A_{t_1-t_0})$, see Definition 13. During interest evaluation, added triples are combined with potentially interesting triples from previous changesets (i.e., $I_{t_1-t_0} = A_{t_1-t_0} \cup \rho_{t_0}$) to check their potential promotion to interesting triples.

Definition 12 (Interest Evaluation over Deleted Triples). *Interest evaluation over deleted triples is a function, $d(i_g, D_{t_1-t_0})$, that returns a 3-element tuple*[5]*:*

$$d(i_g, D_{t_1-t_0}) = \pi(i_g, D_{t_1-t_0}) \ \cup^* \ \pi'(i_g, D_{t_1-t_0}) = \left\langle r_{t_1-t_0}, r_{i(t_1-t_0)}, r'_{t_1-t_0} \right\rangle$$

where:

- $\pi(i_g, D_{t_1-t_0})$ *is an interest candidate generation against deleted triples,*
- $\pi'(i_g, D_{t_1-t_0})$ *is an interest candidate assertion against deleted triples,*
- $r_{t_1-t_0} = \{c_0 \cup c_k \cup c_{op} | c_0, c_k, c_{op} \in \pi(i_g, D_{t_1-t_0})$ *and* $\exists c'_{n-k}, c'_0 \in \pi'(i_g, D_{t_1-t_0})\}$ *is the set of interesting removed triples, i.e., no longer interesting,*
- $r_{i(t_1-t_0)} = \{c_k \cup c_{op} | c_k, c_{op} \in \pi(i_g, D_{t_1-t_0})$ *and* $\nexists c'_{n-k}, c'_0 \in \pi'(i_g, D_{t_1-t_0})\}$ *is the set of potentially interesting removed triples (existing only in removed triples of a changeset) and*
- $r'_{t_1-t_0} = \{c'_0 \cup c'_k \cup c'_{op} | c'_0, c'_k, c'_{op} \in \pi'(i_g, D_{t_1-t_0})$ *and* $\exists c_{op}, c_{n-k}, c_0 \in \pi(i_g, D_{t_1-t_0})\}$ *is the set of triples that become potentially interesting after removing $r_{t_1-t_0}$.*

Example 5. An interest evaluation over deleted triples in our running example (using the results of Example 3 and Example 4, respectively) is as follows:

$$\begin{aligned} d(i_g, D_{05:02-05:00}) &= \pi(i_g, D_{05:02-05:00}) \ \cup^* \ \pi'(i_g, D_{05:02-05:00}) \\ &= \left\langle r_{05:02-05:00}, r_{i(05:02-05:00)}, r'_{05:02-05:00} \right\rangle \end{aligned}$$

1. $r_{05:02-05:00} = c_1$ (in Example 3)

```
dbr:Marcel              dbp:goals    1 .
dbr:Cristiano_Ronaldo   dbo:goals    96 .
```

2. $r_{i(05:02-05:00)} = \varnothing$ (Since all the potentially interesting removed triples of c_1 in Example 3 becomes interesting and no other triples in c_op)
3. $r'_{05:02-05:00} = c'_1$

```
dbr:Marcel              a              dbo:Athlete .
dbr:Cristiano_Ronaldo   a              dbo:Athlete .
dbr:Cristiano_Ronaldo   foaf:homepage  "http://cristianoronaldo.com" .
```

Definition 13 (Interest Evaluation over Added Triples). *Interest evaluation over added triples is a function, $\alpha(i_g, A_{t_1-t_0})$, that returns 3 element tuple as:*

$$\alpha(i_g, A_{t_1-t_0}) = \pi(i_g, I_{t_1-t_0}) \ \cup^* \ \pi'(i_g, I_{t_1-t_0}) = \left\langle a_{t_1-t_0}, a_{i(t_1-t_0)}, a'_{t_1-t_0} \right\rangle$$

where:

[5] Note: $\cup^*$ indicates that after the component-wise union of the two sets the results are combined to three categories of the resulting 3-tuple, namely, (i) elements from left that have matching right elements, (ii) elements from left that do not have matching right elements, and (iii) element from right that have a match left.

- $I_{t_1-t_0} = A_{t_1-t_0} \cup \rho_{t_0}$ *is a set of added triples and potentially interesting triples dataset,*
- $\pi(i_g, I_{t_1-t_0})$ *is an interest candidate generation over* $I_{t_1-t_0}$*,*
- $\pi'(i_g, I_{t_1-t_0})$ *is an interest candidate assertion over* $I_{t_1-t_0}$*,*
- $a_{t_1-t_0} = \{c_0 \cup c_k \cup c_{op} | c_0, c_k, c_{op} \in \pi(i_g, I_{t_1-t_0})$ *and* $\exists c'_{n-k}, c'_0 \in \pi'(i_g, I_{t_1-t_0})\}$ *is the set of interesting added triples,*
- $a_{i(t_1-t_0)} = \{c_k \cup c_{op} | c_k, c_{op} \in \pi(i_g, I_{t_1-t_0})$ *and* $\nexists c'_{n-k}, c'_0 \in \pi'(i_g, I_{t_1-t_0})\}$ *is the set of potentially interesting added triples that do not have related triples in target dataset, and*
- $a'_{t_1-t_0} = \{c'_0 \cup c'_k \cup c'_{op} | c'_0, c'_k, c'_{op} \in \pi'(i_g, I_{t_1-t_0})$ *and* $\exists c_{op}, c_{n-k}, c_0 \in \pi(i_g, I_{t_1-t_0})$ *respectively}* *is the set of triples from target dataset that are related to* $a_{i(t_1-t_0)}$*.*

Example 6. An interest evaluation over added triples in our running example (using the results of Example 3 and Example 4, respectively) is as follows:

$$\begin{aligned}\alpha(i_g, A_{05:02-05:00}) &= \pi(i_g, I_{05:02-05:00}) \ \cup^* \ \pi'(i_g, I_{05:02-05:00}) \\ &= \left\langle a_{05:02-05:00}, a_{i(05:02-05:00)}, a'_{05:02-05:00} \right\rangle\end{aligned}$$

1. $a_{05:02-05:00} = c_1 \cup c'_1 \cup c_0$

```
dbr:Cristiano_Ronaldo   dbo:goals      216 .
dbr:Cristiano_Ronaldo   a              dbo:Athlete .
dbr:Cristiano_Ronaldo   foaf:homepage  "http://cristianoronaldo.com" .
dbr:Rio_Ferdinand       a              dbo:Athlete .
dbr:Rio_Ferdinand       dbp:goals      10 .
```

2. $a_{i(05:02-05:00)} =$

```
dbr:Arvid_Smit     a              dbo:Athlete .
dbr:Barack_Obama   foaf:homepage  "http://www.barackobama.com" .
```

3. $a'_{05:02-05:00} = \varnothing$

Now, we will use the results from Definition 12 and Definition 13 to compute interesting and potentially interesting changesets.

Definition 14 (Interest Evaluation). *An interest evaluation over a changeset* $\Delta(V_{t_1})$ *at time* t_1 *is a function* $e(i_g, \Delta(V_{t_1}))$ *that combines the results from an interest evaluation over deleted triples,* $d(i_g, D_{t_1-t_0})$*, and an interest evaluation over added triples,* $\alpha(i_g, I_{t_1-t_0})$*, to return an interesting changeset and potentially interesting changeset as follows:*

$$e(i_g, \Delta(V_{t_1})) = d(i_g, D_{t_1-t_0}) \quad \chi \quad \alpha(i_g, I_{t_1-t_0}) = \langle \Delta(\tau_{t_1}), \Delta(\rho_{t_1}) \rangle$$

where i_g *is an interest expression over an evolving dataset,* $\Delta(\tau_{t_1})$ *is an interesting changeset (see Definition 15), and* $\Delta(\rho_{t_1})$ *is potentially interesting changeset (see Definition 16).*

Definition 15 (Interesting Changeset). *Let* τ_{t_0} *be a target dataset at time* t_0*. An interesting changeset,* $\Delta(\tau_{t_1})$*, for* τ_{t_0} *at time* t_1 *is defined as:*

$$\Delta(\tau_{t_1}) = \left\langle (r_{t_1-t_0} \cup r'_{t_1-t_0}), \ a_{t_1-t_0} \right\rangle$$

where:

– $r_{t_1-t_0}$ *is the set of interesting removed triples, interesting removed optional triples and potentially interesting removed triples with match found in target dataset during candidate generation,* $\pi(i_g, D_{t_1-t_0})$*,*
– $r'_{t_1-t_0}$ *is the set of triples from target dataset that are related to potentially interesting removed triples computed by* $\pi'(i_g, D_{t_1-t_0})$*, and*
– $a_{t_1-t_0}$ *is the set of interesting added triples, interesting optional triples and potentially interesting added triples with match found in target dataset during candidate generation,* $\pi(i_g, A_{t_1-t_0})$*.*

Example 7. An interesting changeset for our running example is as follows: $\Delta(\tau_{05:02}) = \left\langle \left(r_{05:02-05:00} \cup r'_{05:02-05:00}\right),\ a_{05:02-05:00} \right\rangle$

1. interesting removed triples – $\left(r_{05:02-05:00} \cup r'_{05:02-05:00}\right)$:

```
dbr:Marcel            a             dbo:Athlete .
dbr:Marcel            dbp:goals     1 .
dbr:Cristiano_Ronaldo dbo:goals     96 .
dbr:Cristiano_Ronaldo a             dbo:Athlete .
dbr:Cristiano_Ronaldo foaf:homepage "http://cristianoronaldo.com" .
```

2. interesting added triples – $a_{05:02-05:00}$:

```
dbr:Cristiano_Ronaldo  dbo:goals     216 .
dbr:Cristiano_Ronaldo  a             dbo:Athlete .
dbr:Cristiano_Ronaldo  foaf:homepage "http://cristianoronaldo.com" .
dbr:Rio_Ferdinand      a             dbo:Athlete .
dbr:Rio_Ferdinand      dbp:goals     10 .
```

Triples that were interesting will be downgraded to potentially interesting and stored in ρ_{t_1}, if deletion involves triples matching at least one triple pattern from interest expression BGP.

Definition 16 (Potentially Interesting Changeset). *Let* ρ_{t_0} *be a potentially interesting dataset for interest expression* i_g *at time* t_0*. A changeset,* $\Delta(\rho_{t_1})$*, for* ρ_{t_0} *at time* t_1 *is defined as:*

$$\Delta(\rho_{t_1}) = \left\langle r_{i(t_1-t_0)},\ \left(a_{i(t_1-t_0)} \cup r'_{t_1-t_0}\right) \right\rangle$$

where:
– $r_{i(t_1-t_0)}$ *is a set of potentially interesting removed triples,*
– $a_{i(t_1-t_0)}$ *is a set of potentially interesting added triples computed on added triples of a changeset and related triples extracted from target while removing potentially interesting removed triples, and*
– $r'_{t_1-t_0}$ *is the set of triples from target dataset that are related to potentially interesting removed triples computed by* $\pi'(i_g, D_{t_1-t_0})$*.*

Example 8. Potentially interesting changeset for our running example is as follows: $\Delta(\rho_{05:02}) = \left\langle r_{i(05:02-05:00)},\ \left(a_{i(05:02-05:00)} \cup r'_{05:02-05:00}\right) \right\rangle$

1. Potentially interesting removed triples – $r_{i(05:02-05:00)} = \varnothing$
2. Potentially interesting added triples – $\left(a_{i(05:02-05:00)} \cup r'_{05:02-05:00}\right)$

```
dbr:Arvid_Smit    a             dbo:Athlete .
dbr:Barack_Obama  foaf:homepage "http://www.barackobama.com" .
dbr:Marcel        a             dbo:Athlete .
```

Note: since all triples in $r'_{05:02-05:00}$ are added back to target dataset, they are no longer stored in the potentially interesting dataset.

Definition 17 (Interesting Update Propagation). *An interesting changeset propagation is an update operation that transforms the target dataset τ_{t_0} to the new dataset τ_{t_1} and ρ_{t_0} to new dataset ρ_{t_1} by applying the result of interest evaluation, $e(i_g, \Delta(V_{t_1}))$. That is:*

$$\Upsilon(i_g, \Delta(V_{t_1})) = \upsilon(\tau_{t_0}, \Delta(\tau_{t_1})) \ \wedge \ \upsilon(\rho_{t_0}, \Delta(\rho_{t_1})) = \tau_{t_1} \ \wedge \ \rho_{t_1}$$

- $\Delta(V_{t_1})$ *is a changeset at time* t_1,
- $\upsilon(\tau_{t_0}, \Delta(\tau_{t_1})) = (\tau_{t_0} \backslash [r_{t_1-t_0} \cup r'_{t_1-t_0}]) \cup a_{t_1-t_0}$ *is changeset propagation of interesting changeset, and*
- $\upsilon(\rho_{t_0}, \Delta(\rho_{t_1})) = (\rho_{t_0} \backslash r_{i(t_1-t_0)}) \cup (a_{i(t_1-t_0)} \cup r'_{t_1-t_0})$ *is changeset propagation of potentially interesting changeset.*

Example 9. Propagation of an interesting changeset of Example 7 to the target dataset, τ_{t_0} and potentially interesting changeset of Example 8 to the potentially interesting datasetρ_{t_0} transforms the datasets to:

```
dbr:Cristiano_Ronaldo  dbo:goals      216 .
dbr:Cristiano_Ronaldo  a              dbo:Athlete .
dbr:Cristiano_Ronaldo  foaf:homepage
                    "http://cristianoronaldo.com" .
dbr:Rio_Ferdinand      a              dbo:Athlete .
dbr:Rio_Ferdinand      dbp:goals      10 .
```

Listing (1.3) Resulting target dataset (τ_{t_1})

```
dbr:Arvid_Smit    a              dbo:Athlete .
dbr:Barack_Obama  foaf:homepage
                  "http://www.barackobama.com" .
dbr:Marcel        a              dbo:Athlete .
```

Listing (1.4) Potentially interesting dataset after change propagation (ρ_{t_1})

Fig. 4: State of τ_{t_1} and ρ_{t_1} after propagation

More details on the formalization and implementation of the approach can be found here: `http://eis.iai.uni-bonn.de/Projects/iRap.html`.

3 iRap RDF Update Propagation Framework

In this section we describe the architecture of our interest-based update propagation framework, iRap, and its implementation. iRap was implemented in Java using Jena-ARQ. It is available as open-source and consists of three modules: (1) *Interest Manager* (IM), (2) *Changeset Manager* (CM) and (3) *Interest Evaluator* (IE), each of which can be extended to accommodate new or improved functionality.

Changeset evaluation starts after a user registers an interest expression using the IM service, as shown in Figure 5. The CM module fetches a list of changeset folders from interest expressions and regularly (configurable) checks

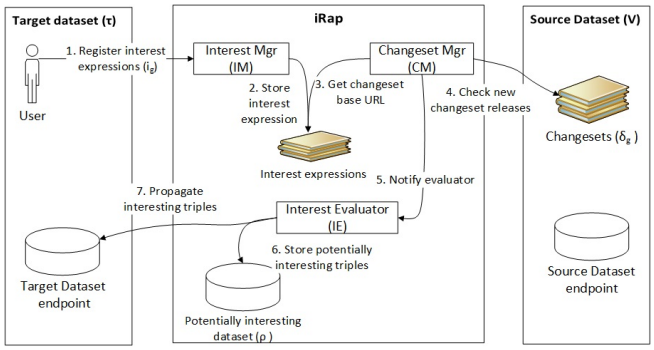


Fig. 5: Architecture of the iRap interest-based RDF update propagation framework.

for new changesets. After downloading and decompressing new changesets, the CM notifies the IE, which then imports a list of interest expressions registered for this particular changeset through the IM and initiates the evaluation. Resulting interesting triples are propagated to the target dataset whereas potentially interesting triples are stored in the potentially interesting dataset (ρ). After all interest expressions have been evaluated over the changeset, the IE notifies the CM to clean the downloaded files.

4 Evaluation

To evaluate the proposed approach, we performed experiments on the iRap framework using changesets published by DBpedia and compared the results with the DBpedia Live Mirror tool. The comparison considers two cases: using iRap to update a previously-established local replica of i) an entire remote dataset ii) a subset of a remote dataset. These two cases simulate two ways in which iRap can be used: i) using interest-based changeset propagation for future updates of a local copy of a large dataset or ii) starting with a new subset of the large dataset.

Experimental Setting In order to test our approach we used the DBpedia dump[6] of September 30, 2014 for the initial setup of the target datasets for two different application domains, namely, *Location and Football* datasets. Changesets published between October 01 and October 15, 2014 were used for evaluation (see Table 1). Changesets are not sequential with modified date but with extraction from DBpedia Live, as discussed in the DBpedia mailing list. Initially we set up two Jena TDB datasets for each target dataset from the DBpedia dump. We loaded all triples from the dump to the Location dataset, whereas for the Football dataset we only loaded a slice corresponding to interesting triples matching Listing 1.5.

[6] `http://live.dbpedia.org/dumps/dbpedia_2014_09_30_00_00.fixed.ttl.gz`

Date	Oct 01	Oct 02	Oct 03	Oct 04-12	Oct 13	Oct 14	Oct 15
Total Changesets	0	1,621	1,755	0	5,352	751	2,578

Table 1: Distribution of DBpedia Live changesets published October 01-15, 2014.

```
CONSTRUCT WHERE {
  ?footballer  a    dbo:SoccerPlayer .
  ?footballer  foaf:name   ?name.
  ?footballer  dbo:team    ?team .
  ?team        rdfs:label  ?teamName.
}
```

Listing (1.5) I_1 – Football interest query

```
CONSTRUCT WHERE {
  ?location  a             ?type .
  ?location  wgs:long      ?long .
  ?location  wgs:lat       ?lat .
  ?location  rdfs:label    ?label .
  ?location  dbo:abstract  ?abstract .
  OPTIONAL { ?location dcterms:subject ?subject }
}
```

Listing (1.6) I_2 – Location interest query

Initially, the Location dataset contains all triples from DBpedia yielding a total of 3 billion triples, whereas the Football dataset contains only 265,622 triples. A total of 12,057 changesets (pairs of removed and added .nt.gz files) have been published in the evaluation timeframe.

The evaluation comprises two interest expressions, I_1 and I_2. I_1 comprises a non-disjoint BGP containing 4 triple patterns with a maximum of two variables per triple pattern (object-subject join), Listing 1.5. I_2 comprises a non-disjoint BGP containing 5 triple patterns with a maximum of two variables per triple pattern (subject-subject joins) and one an OGP containing one triple pattern, Listing 1.6.

We set up two target datasets and potentially interesting dataset using Jena TDB and jena-fuseki for each dataset. The potentially interesting dataset stores potentially interesting triples for each interest expression within a named graph. All experiments were carried out on a 64-bit machine with Windows 7, Intel(R) Core i7-4770 CPU, 16GB RAM and 1TB HD.

Evaluation Results and Discussion Figure 7 summarizes our experimental results for two target datasets shows the growth of the potentially interesting dataset. Results of the interest evaluation for the Football dataset are presented in Table 2. From the overall changesets considered for this evaluation, in Table 1, only 0.38% of the removed and 0.335% of the added triples were identified as interesting for the Football dataset. The average changeset publication interval was 18.81s and average time required for a changeset evaluation is 0.87s. This shows that iRap efficiently performs changeset propagations way before the next changeset is published.

Results of the interest evaluation for the Location dataset are shown in Table 3. From the overall changesets considered for this evaluation, in Table 1, only 4.38% of the removed and 1.81% of the added triples were interesting for the Location dataset. The average time spent for a changeset evaluation is 5.31s. The interest evaluation for the Location dataset takes longer than Football dataset, because of the number of triples in the target dataset was the full DBpedia. Figure 7a shows the number of triples published per day and the number of in-

Day	Total Removed	Interesting Removed	Total Added	Interesting Added	Potentially Interesting	Elapsed (in minutes)
1	1,895,179	9,065	2,051,976	184	169,554	15.18
2	1,748,511	4,865	2,384,232	155	168,856	20.85
3	1,716	0	10,728,855	45,429	684,491	69.86
4	449	0	1,522,939	7,970	97,300	10.17
5	1,677	0	5,234,788	19,598	333,232	60.06

Table 2: Comparison of results for Football App

Day	Total Removed	Interesting Removed	Total Added	Interesting Added	Potentially Interesting	Elapsed (in minutes)
1	1,895,179	77,377	2,051,976	7,093	430376	166.59
2	1,748,511	82,461	2,384,232	7,301	509,972	242.62
3	1,716	0	10,728,855	259,587	2,002,271	417.87
4	449	0	1,522,939	27,292	280,718	64.41
5	1,677	0	5,234,788	100,073	972,284	176.78

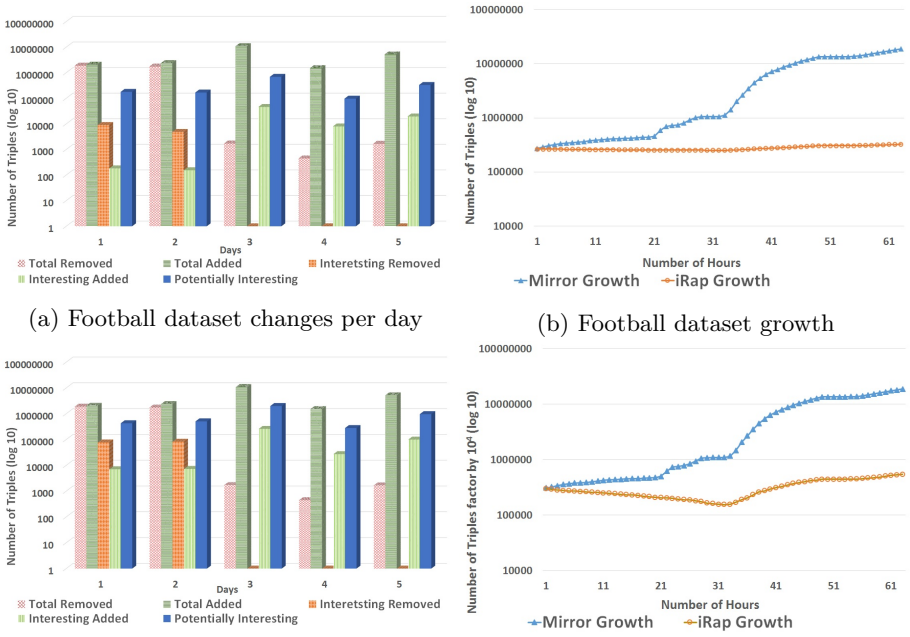
Table 3: Comparison of results for Location App

teresting triples and potentially interesting triples found from interest evaluation for Football dataset. Figure 7b shows the dataset growth comparison between iRap and a full mirror approach. As the figure clearly shows, iRap managed datasets are almost two orders of magnitude smaller and grow much slower than with a mirror approach. Note that the growth for each datasets is calculated by subtracting the number of removed triples from and adding the number of added triples to the total number of triples in the dataset.

We observed a logarithmic growth of the potentially interesting dataset for Location and Football datasets. This is due to the number of variables used in triple patterns, and the number and type of triple patterns in interest expression. For example, the Football dataset interest query contains the common predicates `foaf:name` and `rdfs:label` which are used in almost all resources and thus result in many potentially interesting triples. Again, the average processing time per changeset is always way below the average time between two changesets. The correctness of the resulting triples from the first changesets, for Football dataset interest expression, was checked by manual inspection.

5 Related Work

Most related work on dataset change detection and propagation focuses on distributed publish/subscribe systems [7,3], resource link maintenance [8,10], target synchronization [5], partial replicas [9], data-shipping [12], lazy updates [2], and real-time update notification [10,6]. In [7], the authors propose a peer-to-peer publish/subscribe system for events described in RDF. By avoiding the use of multiple indexes for the same publication they manage to reduce storage space. Similarly, [3] provide an implementation with publish/subscribe capabilities in an RDF-based peer-to-peer system to manage digital resources. As for resource link maintenance, *DSNotify* [8] offers a change-detection framework to detect and fix broken links between resources in two datasets while, *Semantic Pingback* [10] proposes a notification system for the creation of new links between Web re-



(a) Football dataset changes per day

(b) Football dataset growth

(c) Location dataset changes per day

(d) Location dataset growth

Fig. 7: Evaluation results

sources. To note that this approach is suitable for relatively static resources, i.e. RDF documents or RDFa annotated Web pages. In contrast, *SparqlPuSH* [6] offers a real-time notification framework for data updates in a RDF store using a semantic *PubSubHubbub*-based protocol (PuSH). SparqlPuSH allows users to subscribe for changes updates of a subset of content in a RDF store using SPARQL. However, notification and broadcasting are only available as RSS and Atom feeds. As regards target synchronization, *RDFSync* [5] performs update synchronization by merging source and target graphs to get the updated target RDF graph. Alternatively, [9] has designed an approach to replicate, modify, and write-back parts of an RDF graph on devices with low computing power. In distributed databases, where data is replicated on different sites, Lazy update protocols [2] disseminate updates to replicas to ensure consistency. These protocols guarantee serializable execution as well as high performance.

6 Conclusion and Future Work

In this paper we presented a novel approach for interest-based RDF update propagation that can consistently maintain a full or partial replication of large LOD datasets. We have demonstrated the validity of the approach through detailed formalizations and their application in a reference implementation of the iRap Framework. A thorough evaluation of the approach indicates that our method

can significantly cut down on both the size of the data updates required to consistently maintain a localized dataset replication up-to-date, as well as the speed by which such updates can take place. Future work will focus on extending the iRap Framework with a publish/subscribe distributed architecture. The framework will be improved also from the usability point of view, including a user interface and making the initial generation of RDF slices easier and more efficient. An extensive evaluation of scalability and performance of the framework will be performed and a benchmark dataset for future reference will be made available to the research community.

Acknowledgments: This work is supported by LinDA project, funded by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement number 610565.

References

1. SPARQL 1.1 Query Language. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (2013)
2. Breitbart, Y., Komondoor, R., Rastogi, R., Seshadri, S., Silberschatz, A.: Update propagation protocols for replicated databates. ACM SIGMOD Record 28 (1999)
3. Chirita, P., Idreos, S., Koubarakis, M., Nejdl, W.: Publish/subscribe for rdf-based P2P networks. In: 1st European Semantic Web Symposium ESWS (2004)
4. Marx, E., Shekarpour, S., Auer, S., Ngomo, A.N.: Large-scale RDF dataset slicing. In: 2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013. pp. 228–235 (2013)
5. Morbidoni, C., Tummarello, G., Erling, O., Bachmann-Gmür, R.: Rdfsync: efficient remote synchronization of rdf models. In: 6th ISWC and 2nd ASWC 2007. LNCS, vol. 4825, pp. 533–546. Springer (2007)
6. Passant, A., Mendes, P.: sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. In: Scripting for the Semantic Web Workshop (SFSW2010) (2010)
7. Pellegrino, L., Huet, F., Baude, F., Alshabani, A.: A distributed publish/subscribe system for RDF data. In: Data Management in Cloud, Grid and P2P Systems - 6th Int. Conf., Globe 2013. pp. 39–50 (2013)
8. Popitsch, N., Haslhofer, B.: Dsnotify - A solution for event detection and link maintenance in dynamic datasets. J. Web Sem. 9(3), 266–283 (2011)
9. Schandl, B.: Replication and versioning of partial rdf graphs. In: ESWC (1). Lecture Notes in Computer Science, vol. 6088, pp. 31–45. Springer (2010)
10. Tramp, S., Frischmuth, P., Ermilov, T., Auer, S.: Weaving a social data web with semantic pingback. In: EKAW 2010. LNAI, vol. 6317, pp. 135–149. Springer (2010)
11. Verborgh, R., Hartig, O., De Meester, B., Haesendonck, G., De Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., Van de Walle, R.: Querying datasets on the web with high availability. In: ISWC 2014, pp. 180–196. Springer (2014)
12. Voruganti, K., Özsu, M.T., Unrau, R.C.: An adaptive data-shipping architecture for client caching data management systems. Distributed and Parallel Databases 15(2), 137–177 (2004)