DELTA-R:
A Change Detection Approach for RDF Datasets

Anuj Singh, Rob Brennan and Declan O'Sullivan
ADAPT Centre, School of Computer Science and Statistics, Trinity College Dublin, Ireland
{singh.anuj, rob.brennan, declan.osullivan}@adaptcentre.ie

Abstract. This paper presents the DELTA-R approach that detects and classifies the changes between two versions of a linked dataset. It contributes to the state of the art *firstly:* by proposing a more granular classification of the resource level changes, and *secondly:* by automatically selecting the appropriate resource properties to identify the same resources in different versions of a linked dataset with different URIs and similar representation. The paper also presents the DELTA-R change model to represent the changes detected by the DELTA-R approach. This model bridges the gap between resource-centric and triple-centric views of changes in linked datasets. As a result, a single change detection mechanism will be able to support the use cases like interlink maintenance and dataset or replica synchronization. Additionally, the paper describes an experiment conducted to examine the accuracy of the DELTA-R approach in detecting the changes between two versions of a linked dataset. The result indicates that the accuracy of DELTA-R approach outperforms the state of the art approaches by up to 4%. It is demonstrated that the proposed more granular classification of changes helped to identifyup to *1529 additional updated resources* compered to X.By means of a case study, we demonstrate the support of DELTA-R approach and change model for an interlink maintenance use case. The result shows that 100% of the broken interlinks were repaired between DBpedia person snapshot 3.7 and Freebase.

Keywords: Change detection, link maintenance, dataset dynamics, linked data

1 INTRODUCTION

Many linked datasets are highly dynamic in nature [8]. For an application consuming a dynamic linked dataset, the dynamic nature of the dataset may result in issues for the application such as broken interlinks or outdated data [1].

These issues are typically encapsulated in the research community using the term "dataset dynamics". Dataset dynamics investigates the approaches that deal with changes in linked datasets at different levels of granularity (triple, resource, or graph-level), during the evolution of the datasets [2]. The aim is to build: (i) vocabularies to represent the change information, (ii) mechanisms for change detection, and (iii) change propagation methods. Change detection in linked datasets has proven to be important for supporting use cases like interlink maintenance and synchronization of dataset versions, replicas, and interconnected datasets [1, 5]. However, existing change detection mechanisms have certain limitations in our opinion (see below).

These limitations have motivated our research in dataset dynamics: (a) **Classification granularity limitation** – the existing classification of changes does not distinctly identify all the resources that changed their representation and may lead to semantically broken interlinks. An interlink is semantically broken when the meaning of the representation of source and target resources differs from each other; (b) **Properties selection limitation** – the existing approaches either use *specific properties* or all the *object properties* (graph structure) of a resource to identify the resources that have changed their address and possibly their representation (structure and/or values) too. Identification of specific properties is generally based on its entropy or coverage in the dataset, the calculation of which requires an additional effort (pre-change detection) [1]. The reliance on object properties may give good recall, but the precision could be severely affected, as some of the object properties are highly generic in nature, the argument is supported by the results in [6]; (c) **Change model**[1] **limitation** – the existing change models do not allow one to represent the changes such that one can access a resource level change along with its corresponding changed triples. We believe that the existence of this gap between resource centric view and triple centric view of changes in linked dataset restricts a single change detection mechanism to support use cases like interlink maintenance, dataset or replica synchronization.

The specific *research question* under evaluation in this paper is to what extent we can detect and classify the resource level changes between two versions of a linked dataset.

The contribution of this paper is as follows: *Firstly*, it proposes DELTA-R, a novel change detection and classification approach for linked datasets. DELTA-R addresses the limitations in the state of the art **(a)** and **(b)** described above. *Secondly*, it proposes the **DELTA-R change model** to represent the changes identified by the DELTA-R approach, it addresses limitation **(c)**. *Thirdly*, it evaluates the accuracy of the DELTA-R approach using real world data compared

[1]The generic model to represent changes. This model can be incorporated in ontology, XML schema, or any other vocabulary.

to state of the art change detection approaches. *Fourthly*, it demonstrates how the changes detected can be used to support a structurally broken interlink repair use case.

The paper is organized as follows: **Section 2** discusses how the aforementioned limitations exist in state of the art approaches. **Section 3** describes the proposed DELTA-R approach. **Section 4** describes the DELTA-R change model proposed. **Section 5** presents an evaluation of DELTA-R in terms of accuracy achieved. **Section 6** discusses a case study that was performed to fix structurally broken interlinks using the change information generated by DELTA-R. Conclusions are drawn in **Section 7**.

2 RELATED WORK

Popitsch and Haslhofer [1] proposed three representative use cases where applications consuming linked dataset need to be informed about the changes in a consumed dataset. **Use Case 1a - Semantic link maintenance** – in this, the applications need to be informed about the resources that have changed their representation. **Use Case 1b - Structural link maintenance** – in this, the applications need to be informed about the resources that have changed their address (Subject URI) or have been deleted from the dataset. Use cases 1a and 1b need the information about changed resources thus, what is called *resource level change information* is required to support these use cases. **Use Case 2 - Dataset synchronization** – in order to synchronize a replica of a linked dataset, it is important to identify the triples that have been deleted or added in the main dataset. **Use Case 3 - Data caching** – applications consuming remote linked dataset(s) maintain a http cache to store remote data locally [1]. To synchronize the locally stored data, it is again important to identify the triples that have been deleted and added in the remote datasets. Use case (2) and (3) need the information about the changed triples, thus, what is called *triple level change information* is required to support these use cases. The approaches that support these use cases are discussed in this section. Table 1 summarizes the categorization of the state of the art approaches according to their target use case(s) and level of change information required.

Table 1. Categorization of the SOA approaches according to their target use case and change information level required.

Approaches	Change Information Level	Interlink Maintenance		Dataset Synchronization	Data Caching
		Structural	Semantic		
Popitsch et al. [1]	Resource	X			
Pourzaferani et al. [6]	Resource	X			

Kovilakath et al. [3]	Resource		X		
Pernelle et al. [7]	Triple			X	X
Lee et al. [10]	Triple			X	X

Approaches proposed in [1,3, and 6] identify the change information at resource level. The approach in [3] detects semantically broken interlinks by keeping track of updated resources using PubSubHubbub. However, this approach is only able to partially detect semantically broken interlinks, as the approach is not able to detect the updated resources that also have changed its subject URI at the same time. The approach in [6] detects the resources that changed its subject URI to identify and repair structurally broken interlinks. However, this approach is not able to cater for all structurally broken interlinks as this approach does not identify removed resources. To identify all the possible candidate resources that can lead to structurally broken interlinks, the approach in [1] detects and classifies the changes in linked datasets during its evolution. To the best of our knowledge the approach provides the most granular classification of changes in linked datasets at *resource level*. There are four types of changes that exist in this classification:

a) *Create* – addition of a new resource in linked dataset;

b) *Remove* – deletion of an existing resource from the dataset;

c) *Update* – change in representation of an existing resource in the dataset;

d) Move – change in the subject URI (address) of an existing resource in the dataset. Also, it is possible that the representation of the resource has also been changed. Using the remove and move type of changes, one can identify all the potential candidate resources that can lead to structurally broken interlinks. [1] did not intend to support the semantically broken interlink maintenance use case, but the classification in itself is able to support this use case by the identification of updated resources. However, not all of the updated resources can be identified using this classification. This is because this classification does not differentiate between the resources that changed their address and the resources that changed both their address and representation at the same time. This limitation in classification granularity has been termed in the introduction section as limitation (a).

Out of all the resource level change detection approaches discussed above, only [1 and 6] can identify moved resources. However, [1] requires configurable properties, while [6] relies on object properties for the identification of moved resources. This limitation of property selection has been referred in the introduction as limitation (b).

To identify the triple level change information, [7] proposes an approach of change detection for evolving RDF datasets. This approach identifies the triples that have been deleted and added in the dataset during its evolution. Another approach in [10], also identify triple level change information in RDF datasets.

This approach considers the existence of blank nodes in RDF datasets. The approaches in [7 and 10] identify the triple level change information, thus, are suitable to support use cases 2 and 3.

In this section so far, we have identified that the state of the art approaches detect changes at two levels, triple and resource, with the different levels supporting different use cases. To the best of our knowledge, the existing approaches do not detect and represent change information at both resource and triple levels.

Out of the approaches discussed above only [1] and [7] proposed vocabularies to represent the detected changes in RDF or linked datasets. [7] proposed an ontology O^{DE} to represent triple level changes. In O^{DE}, each reified triple that belongs to either added or deleted set of triples detected between two versions of an RDF dataset, is a class instance. In the reification of triples, the proposed approach, adds the type of each added or deleted triple. This type also includes the addition, deletion, and enrichment (update) of an instance. This reification bridges the gap between resource (instance) and triple centric view of changes. However, we believe it is not a very effective approach, as a resource (instance), constitutes multiple triples, so, this reification will generate redundant information about the resource level changes. On the other hand, [1] provides a vocabulary to represent the changes in linked dataset at resource level. The vocabulary by [1] also has a "hasAffectedTriples" property that can be used to specify the affected triples due to the changes in a resource. This property bridges some of the gap between the resource and triple centric view of changes in linked datasets. However, the vocabulary still does not provide a way to distinctly identify the added and deleted triples due to the changes in the representation of a resource. This limitation in representing the resource and triple level changes together has been referred in the introduction as limitation (c).

We have shown that limitations mentioned in the introduction exist in state of the art approaches. We argue in this paper that in order to address these limitations we need to: **(a)** separate in a classification of resources that only change their address and resources that both change their address and representation at the same time, i.e. sub classify the "move" type of change; **(b)** a change mechanism that does not require pre-determined properties for the identification of moved resources; **(c)** a change model that bridges the gap between resource and triple centric view of changes.

3 The DELTA-R Approach

The main objective of the DELTA-R approach is to address the limitations (a) and (b) of current approaches. To address limitation (a), DELTA-R proposes a new

classification for resource level changes in linked datasets. The proposed classification has the following types of changes: *Create* – addition of a resource in a linked dataset; *Remove* – removal of a resource from a linked dataset; *Update* – change in the representation of an existing resource of a linked dataset; *Move* – change in the subject URI(address) of an existing resource; **Renew** – **change in the address as well as in the representation of an existing resource**. To address limitation (b), the approach first identifies all the potential moved and renewed resources using all the properties present in the resources. Then to filter out the incorrect moved or renewed resources, the approach determines the critical properties (section 3.3) automatically.

The DELTA-R approach contains four major activities (see Figure 1). First is the **Ingestion activity** (section 3.1) – it uploads both the older and newer versions of a linked dataset into a triple store. Second is the **Feature extraction activity** (section 3.2) –this uses the earlier uploaded versions of the dataset to extract the properties and object information of new and deleted resources. Third is the **Change detection and classification activity** (section 3.3) – it uses the extracted properties and objects of new and deleted resources to identify the updated, moved and renewed resources. Fourth is the **Transformation activity** (section 3.4) – it transforms the identified change information into RDF using the DELTA-R change model vocabulary (explained in section 4).

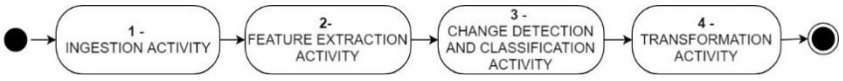

Figure 1: Overview of the major activities in the DELTA-R approach

In the following sub-sections, each activity of DELTA-R approach in described in more detail.

3.1 INGESTION ACTIVITY

Requirements: The aim of this activity is to upload the RDF dataset dumps in a triple store, in a way that the different dumps can be distinctly identified inside the triple store.

Implementation: we have implemented an Ingestion component which uploads RDF dataset dump for both older and newer versions in a triple store, in two distinct graphs. We have used the *sem:rdf-load()* function of the semantic API of the MarkLogic platform to upload the RDF dataset dumps in the MarkLogic

triple store[2]. First, the older version, then the newer version was uploaded, in two distinct graphs.

3.2 FEATURE EXTRACTION ACTIVITY

Requirements: The aim of this activity is to extract every property and object of each newly added and deleted resource from the uploaded datasets. The extracted information is required to generate keys that can be used for the change detection mechanism. This is important to identify the same resources with different subject URI and similar representation (but not the same).

Implementation: We implemented a Feature Extraction Component that extracts the properties and objects of each newly added and deleted resource in the uploaded dataset versions. The aim is to create and store a separate representation of the extracted properties and object in the MarkLogic database[3]. The subject URI of a resource is also attached in its separate representation. This representation will be utilized at the change detection stage. Figure 2 provides a brief description of all the steps performed by this component.

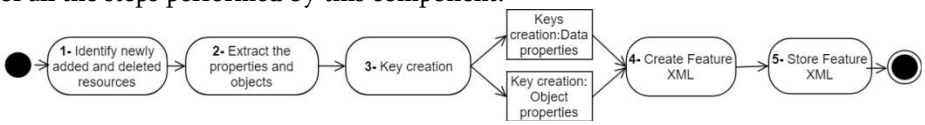

Figure 2. Sequence of steps of the Feature extraction component

Step 1, SPARQL queries described in Figure 2 are used to identify the newly added and deleted resources between the older and newer version of the dataset.

```
SELECT DISTINCT ?S
WHERE
{
  GRAPH <$newer-version>
  {  ?S ?P ?O  }
  FILTER NOT EXISTS
  {
    GRAPH <$older-graph>
    { ?S ?P ?O }
  }
}
```

(a) identify newly added resources

```
SELECT DISTINCT ?S
WHERE
{
  GRAPH <$older-version>
  {  ?S ?P ?O  }
  FILTER NOT EXISTS
  {
    GRAPH <$newer-graph>
    { ?S ?P ?O }
  }
}
```

(b) identify deleted reosurces

Figure 2. SPARQL queries used to identify newly added and deleted resources

Step 2, The component then extract the properties and object of the identified resources from earlier uploaded dataset versions and additional information[4] that

[2]https://www.marklogic.com/product/marklogic-database-overview/database-features/semantics/

[3]https://www.marklogic.com/

[4]This additional information could be provenance, archival mementos, etc.

may exist. Before executing this step, the additional information (for both older and newer version of the dataset) is uploaded in the triple store, in distinct graphs. It is an optional step. *Step 3*, to cope with the potential errors generally made by humans at data entry stage [9], algorithm transforms the object values (from triples in the earlier uploaded versions and additional information) of the identified resources into keys. The created keys will be used to calculate the similarity between two resources in different versions. The keys are formed differently for different types of object values:

– *URIs (object properties that contains no numeric character)*: The algorithm uses the last token (tokenize with '/') of URI path as the key;

– *text (data properties that contains no numeric character)*: only the first three tokens (using \s) of the strings are used for the key creation. The algorithm combines the sub-keys from each token to form a single key. To create the sub-key of each token, the algorithm takes all the distinct vowels in a token, first and last character of the token, and the primary key of double metaphone[5] encoding for the token; For an instance, be "Hamid" a single token object value, the key for this will be "aihhmtd", where "ai" are distinct vowels, "h" is the first character, "hmt" is the primary key of the metaphone encoding, and "d" is the last character of the object. The combination of vowels, first and last characters, and phonetic encoding has been used to improve the accuracy of the algorithm while calculating the similarity of two resources [12].

– *Numbers/ string with digits*: the algorithm uses the exact value of the objects as key. *Step 4*, the generated keys are combined with their property/ predicate name into a **feature** and the XML representation of the features is known as **Feature XML**. A snippet of Feature XML is shown in Figure 3.

```
<allFeatures res="http://dbpedia.org/resource/Hamid_Taqvaee" state="delete">
      <givenname type="text">aihhmtd</givenname>
      <name type="text">aihhmtdaettkfe</name>
      <surname type="text">aettkfe</surname>
      <subject type="numeric">1949_births</subject>
      <subject type="uri">sharif_university_of_technology_alumni</subject>
      <subject type="uri">worker-communist_party_of_iran_politicians</subject>
</allFeatures>
```

Figure 3. Feature XML for a resource key

Step 5, MarkLogic database is used to store the Feature XMLs. The algorithm stores the Feature XMLs of the newer and older resources in "new" and "delete" collections[6] respectively. At this stage, there is no separate collection for the updated resources.

[5]http://www.b-eye-network.com/view/1596

[6]https://docs.marklogic.com/guide/search-dev/collections

3.3 CHANGE DETECTION AND CLASSIFICATION ACTIVITY

Requirement: We need an activity to: **(a)** classify updated resources; – **(b)** identify resources that have only changed their address and classify them as *move* – **(c)** identify resources that have changed both their address and representation at the same time and classify them as *renew*.

Implementation: To fulfill requirement (a), the component identifies the updated resources, their corresponding added and deleted triples and stores all the identified information in "update" collection. To achieve this, the component performs the following steps: *Step 1:* For an updated resource, a Feature XML must be present in both new and delete collection. Feature XML in the delete and new will be denoted by older and newer XML respectively in this section. The component iterates over older XMLs to get the subject URI attached to them. Step 2: If a newer XML has the same subject URI, the component extracts all the triples of this subject URI from older and newer version of the dataset uploaded earlier. Step 3: Then the component compares the extracted triples to identify the added and deleted triples. Step 4: The component then stores the information of the subject URI along with their added and deleted triples in "update" collection. Finally, the older and newer XML gets deleted from delete and new collection respectively.

For requirement (b), the component identifies the moved resources by matching the remaining older XMLs with the remaining newer XMLs. A pair of older and newer XML is decided as match, when older and newer XML shares similar features. The component incorporates three configuration parameters to facilitate matching: - *accept threshold:* a match having confidence value[7] equal or above this threshold is selected as an authentic match; - *audit threshold:* a match having confidence value above this threshold and lower than the accept threshold goes to the audit routine, which decides the authenticity of the match; - *critical feature threshold:* in the context of DELTA-R, a feature is critical if its rate of being same in matches (confidence value > accept threshold) is greater than the critical feature threshold. The critical features are used by audit routine to decide the authenticity of a match. Finally, the component stores the information of the authentic matches in "move" collection. Figure 4 provides a brief description of all the steps performed by this component.

[7] The confidence value is calculated by the following formula: ((no of features matched between the older and newer XML) / (no. of features in the older XML)) * 100

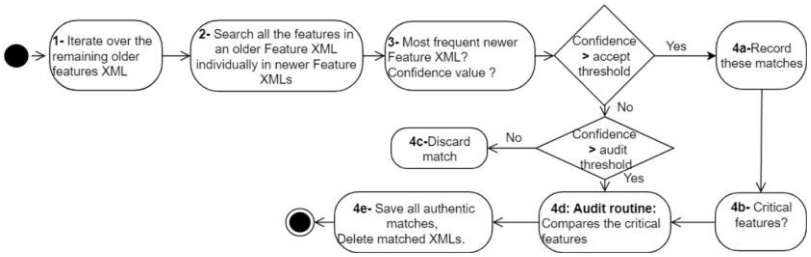

Figure 4: Requirement (b) - identification and classification of moved resources

Step 1: The component iterates over the remaining older XMLs. *Step 2:* Then, the component searches for a similar newer XML by searching all the features in an older XML individually in the newer XMLs. Each older feature can be found in more than one newer XML. The component keeps the listing of all the newer XMLs that appeared in the search of all features of an older XML. *Step 3:* The newer XML with the highest occurrences in the listing will get matched, i.e. the resource URI attached to both older and newer XML will be kept as a pair for further processing and will be denoted by "moved resources" in this paper. Next, the component calculates the confidence value of the identified match. *Step 4a:* If the confidence value is greater than the accept threshold, the match is authentic. The component records these matches in memory for next step. *Step 4b:* Next, the component identifies the critical features by comparing each feature of the matched XMLs of the recorded matches. *Step 4c, 4d:* Subsequently, to decide the authenticity of the matches having confidence value between accept and audit threshold, the audit routine ensures that the critical features are same in the matched XMLs. The component discards all the matches having confidence value lower than the audit threshold. *Step 4e:* The component stores the information of moved resources corresponding to all the authentic matches in "move" collection. Finally, the component deletes all the matched XMLs from the delete and new collection.

For requirement (c), the component identifies the renewed resources by checking if the identified moved resources have a difference in their representation. To achieve this, the component performs the following steps: *Step 1:* The component iterates over each moved resource's information stored in "move" collection. *Step 2:* the component extracts the properties and objects of the moved resources from the older and newer version of the dataset and compares them. *Step 3:* For all the moved resources which has an added or deleted property or object, the component transfers the information of these moved resources from the "move" to "renew" collection. Otherwise, the component does not perform any operation.

The change detection and classification component classifies the changed resources into *updated*, *moved* and *renewed* resources. The resources related to

the remaining Features XMLs in delete and new collection are the *deleted* and *created* resources respectively.

3.4 TRANSFORMATION ACTIVTY

Requirement: The aim is to represent the information of the classified changes in RDF that follows DELTA-R change model (explained in section 4).

Implementation: We have designed a transformation component to implement this activity, which performs the following steps: To achieve this, the component performs the following steps: *Step1:* The component sequentially accesses the information in each new, delete, update, move, and renew collection. *Step 2:* The component transforms the XML information in each collection to RDF that follows the DELTA-R change model. Step 3: Finally, the component stores the transformed information in distinct graphs. To create the distinct graph URIs, the component suffixes a predefined string (http://marklogic.com/semantics/changes/) with the type of change.

4 DELTA-R CHANGE MODEL

To represent the changes identified by DELTA-R approach, we propose the **DELTA-R Change Model**. This model addresses limitation (c) mentioned in the introduction. The DELTA-R model is based on the model proposed in [11]. [11] has proposed a Layered Log Change Model for representing ontology changes. This model contains two levels of granularity: *first level* - represents the information of change operation at atomic level; *second level* – represents the objective of the atomic change. A similar multi-level model can be applied to represent the changes of linked datasets, which in result will bridge the gap between the resource and triple centric view of changes. Analogous to [11], we can represent the deleted and added triples at the atomic (first) level, while at the second level the objective of deleting or adding a triple can be represented as the creation, removal, update, movement, or renewal of a resource. Figure 6 describes DELTA-R change model. This is a generic model that can be incorporated in ontologies, XML schema, or any other vocabulary.

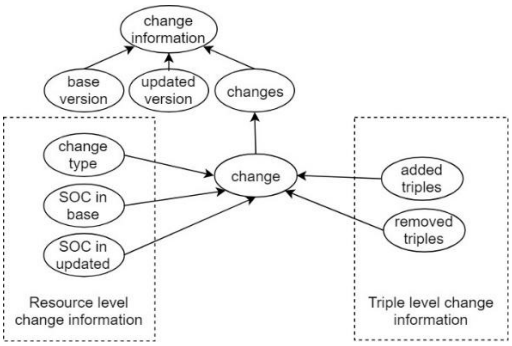

Figure 6: DELTA-R change model

In Figure 6, the "base version" and the "updated version" entities are the older and newer version of a linked dataset used for change detection; the "change type" entity represents the type of resource level change i.e. create, remove, update, move, renew; "SOC in base" and "SOC in updated" entities represents the subject of change (resource URI) in base and updated version respectively; finally, the "removed triples" and "added triples" entities are used to represent the information of the deleted and added triples respectively. Using such change model, one can identify the resource level changes in linked datasets along with their corresponding added and deleted triples.

5 EVALUATION – Accuracy of DELTA-R

The purpose of the experiment was to determine the accuracy of DELTA-R approach and compare with existing approaches [1 and 6]. To the best of our knowledge only [1 and 6] use the similar change metrics as DELTA-R, which is why these approaches have been selected for the evaluation. The hypothesis was that the accuracy of DELTA-R approach would be better than [1 and 6] in terms of F-measure.

Datasets: The experiment has been conducted using two different sets of input. Single set of input contains, two versions of a linked dataset, additional information datasets, and a gold standard to determine the accuracy of DELTA-R approach.

First set: For change detection, we have used the enriched DBpedia person snapshots 3.2 (**20,284** resources) and 3.3 (**29,498** resources) provided by [1]. For the resources in snapshot 3.2 and 3.3 we have used the additional information present in the article category dataset 3.2[8] and 3.3[9] respectively. The following are the

[8] http://oldwiki.dbpedia.org/Downloads32#articlescategories

[9] http://oldwiki.dbpedia.org/Downloads33#articlescategories

reasons to include article category datasets as additional information: - to demonstrate that DELTA-R approach is easily extensible to include additional information; - to provide a wider range of properties to DELTA-R approach for determining critical features; - [6] have also used the article category dataset for the identification of moved resources. So, including the same dataset in our change detection process will give us an opportunity to discuss the impact of critical features on the results.

To determine the accuracy of DELTA-R approach, we used the gold standard provided by [1]. The gold standard contains 179 move type of changes. During the analysis of our results, we found 1 move resource[10] that is not covered by the gold standard as move type of change. Hence, we increased the move type of changes in gold standard by 1. Also, **5666** resources were excluded by [1] for detecting changes. These resources are mentioned as "unknown-created" and "unknown-removed" in the gold standard. In order to have the same baseline, we omitted these resources for change detection. The used datasets and the gold standard are available online[11].

Second set: We have also applied DELTA-R approach on DBpedia person snapshot 3.6[12] (**296,595** resources) and 3.7[13] (**790,703** resources). These datasets are much bigger than the datasets in the first set. Again, for the additional information, we have used the article category dataset of the corresponding DBpedia versions. To the best of our knowledge, the information of gold standard for the changes between these versions is not available in the community. Thus, to determine the accuracy of DELTA-R, we created a gold standard for the resources that changed their address or both address and representation. The Gold standard has been created in following three steps: *Step 1*: We used DBpedia redirect dataset version 3.7[14] and extracted the redirects in which the source URI is present in person snapshot 3.6 and target URI is present in the person snapshot 3.7. By doing this, we were able to identify **3390** redirects. These redirects can be treated as the resources that have changed their address. *Step 2:* During the analysis of our results we found some of the detected move and renew type of changes in DBpedia disambiguation dataset version 3.7 [15]. In DBpedia disambiguation dataset 3.7, some of the resources URIs of person snapshot 3.6 are linked with one or more different resources URIs of the person snapshot 3.7. However, there is only a single link between older resource URI and newer

[10] Older: http://dbpedia.org/resource/Kim_Jin-Kyu; newer: http://dbpedia.org/resource/Kim_Jin-Kyu_%28football_player%29

[11] http://www.cibiv.at/~niko/dsnotify/Download.html#pub

[12] http://oldwiki.dbpedia.org/Downloads36#persondata

[13] http://oldwiki.dbpedia.org/Downloads37#persondata

[14] http://oldwiki.dbpedia.org/Downloads37#redirects

[15] http://oldwiki.dbpedia.org/Downloads37#disambiguationlinks

resource URIs, which denotes that both older and newer resource are the same, it is just that the older resource has been moved to a different address(URI). We have filtered out **585** this type of links from the disambiguation dataset to prepare the gold standard. *Step 3:* Finally, we have manually verified the move and renew types of changes from our results, which are not present in the gold standard prepared in Step 1 and 2. In the verification, we found **296** instances of move and renew type of changes that are correctly detected by DELTA-R approach.

Experimental method: We conducted the experiment on a machine having 7th generation i7 processor with 16 GB RAM. The experiment has been conducted in two stages. *First stage* – the DELTA-R approach has been applied to the datasets of the first input set. For this, we uploaded the person snapshot 3.2 and 3.3 in MarkLogic triple store, in two distinct graphs. We then uploaded the additional information datasets in MarkLogic triple store, in two other distinct graphs. Next, we extracted and stored the features of the resources in the person snapshot 3.2 and 3.3, by using the functionality of the feature extraction component (explained in section 3.2). Once the features have been created in the new and delete collection, the change detection mechanism has been invoked using the following three configurations: accept threshold – 80%; audit threshold – 40%; critical feature threshold – 98%. This resulted in classified changes between person snapshot 3.2 and 3.3. For determining the accuracy of the detected changes, the results were first compared with the gold standard, and the extra changes detected by DELTA-R were manually verified. *Second stage* – the same steps from first stage have been executed.

Results: Table 2 describes the classification of changes detected by DELTA-R approach for both sets of input. For first set of input, we identified 3820 resources that were newly added in the person snapshot 3.3 and 240 resources that were part of the snapshot 3.2 but were not included in the snapshot 3.3. It has also been identified that the representation of 4161 resources have been changed from the snapshot 3.2 to snapshot 3.3. Having these 4161 resources analyzed, we observed that the representation of these resources was changed in three ways: - change in the object value of the existing triples of a resource; - addition of new properties in a resource; - deletion of existing properties of a resource. Additionally, we have identified 124 resources that were using a different subject URI in the snapshot 3.2 than in snapshot 3.3. Furthermore, we detected 45 renewed resources. These are special type of resources because the subject URI and the representation of these resources have changed from the snapshot 3.2 to 3.3. These 45 resources were detected with the help of the proposed new classification of the resource level changes in linked datasets, as existing classifications do not distinctly identify renewed resources. Hence, we can say with the proposed classification of changes, we were able to identify 45 additional resources that changed their representation may lead to semantically broken interlinks. Similarly, we have detected and

classified the changes for second set of input. As the second set of input is much wider than the first input set, thus, the number of detected changes for second input set are much greater than the changes detected for the first set of input. For second set of input, in comparison to the existing classification of changes, we identified 1529 additional resources that changed their representation and may lead to semantically broken interlinks.

Table 2: Detected changes for first and second set of input

Set of input	Create	Remove	Update	Move	Renew
First	3819	239	4161	124	46
Second	499590	5482	50380	2723	1529

To determine the accuracy of DELTA-R approach, we only compared the move and renew types of changes with gold standard. This is because the state of the art emphasizes on determining the accuracy for detecting the resources that have changed their address and possibly their representation too. Since the gold standard does not cater move and renew types of changes separately, we have merged the detected move and renew types of changes as move type of changes. Move and renew types of change will be referred to as the move type of change in this section from now onwards. Table 3 describes the accuracy of DELTA-R approach in detecting move type of changes. For the first set of input, the precision of the DELTA-R approach is 1, this is because the approach ensures that the critical features are same in the low confidence matches. For the recall, the approach was not able to detect 10 moved resources. Out of 10, 5 moved resources got rejected by the audit routine as the match (pair of older and newer resource's feature XML) did not have the same critical features. Rest 5 moved resources had one to many matches, i.e. one older resource was matched with more than one newer resources. Hence, the DELTA-R approach decided to discard these matches. Overall the calculated F-measure for the first input set is 0.9714.

We observed that the precision has come down for the second set of input. The incorrect move type of changes exist in two forms: incorrect move with higher confidence value; incorrect move with lower confidence value. We have analyzed these incorrect move type of changes and found that the majority (~ 50%) of the these belongs to the move with lower confidence value (< 50). Move with such low confidence value were not identified for the first set of input. This suggests that the audit threshold needs to go slightly up, i.e. from 40 to 50%. Doing so, the precision will increase by ~2% with a slight impact on the recall i.e. a decrease in recall by ~0.1%. We have also investigated the incorrect move with higher confidence value and found that the older and newer XML shares high percentage of same features, which include the critical features as well. For the second set of input the calculated recall is greater than the recall for the first input set. We

believe that the reason for this would be the presence of the wider range of information about the resources in the second input set, which allowed the DELTA-R approach to have more one to one matches between older and newer resource's Feature XML.

Table 3: Accuracy of DELTA-R in detecting moved resources

Set of input	Move in gold standard	Move by DELTA-R	Precision	Recall	F-measure
First	180	170	1	0.9444	0.9714
Second	4271	4252	0.9597	0.9555	0.9579

[1 and 6] have also evaluated their system using the first set of input. To identify the moved resources [1] performed the experiment using various configuration properties. The maximum recorded precision and recall were using the foaf:name property, i.e. the precision was 1.0 and the recall was ~0.91. Their results showed the decrease in the F-measure with the increase in the events i.e. the increase in the number of resources for matching. The maximum recorded f-measure for foaf:name was ~0.95. The other approach in [6] has recorded its precision ~0.87 and the recorded recall was ~0.99. The approach has used only the object properties (graphical structure of resources) to identify the moved resources, which we believe is the reason for the lower precision and extremely good recall. The recorded f-measure by [6] was ~0.93. We had also included the same object properties through additional information for the creation of features. However, deciding the critical features out of all the features helped us in achieving more precise results. By comparing our results of first input set with the aforementioned results of the approaches [1 and 6], it has been identified that the DELTA-R approach outperforms [1] by ~2% and [6] by ~4% in terms of F-measure. Hence the results are in support of the hypothesis. We were not able to compare the results of the second input set with any other approach, as to the best of our knowledge, only [6] has published their results of the detected moved resources between DBpedia person snapshot 3.6 and 3.7. But the gold standard used by [6] has a different count than the gold standard used by us. [6] neither contains the information of the creation of their gold standard precisely, nor they have published their gold standard. This refrains us to compare our results of second input set with other approaches. For evaluation of the change detection approaches to be proposed in future, the gold standard prepared by us for the moved and renewed resources between DBpedia person snapshot 3.6 and 3.7 is available online{LINK}.

6 CASE STUDY: Repair of broken interlinks of DBpedia

In this case study, we demonstrate that the classification of changes by DELTA-R approach and their representation using DELTA-R change model can be utilized to repair the structurally broken interlinks automatically. For this, we have repaired and validated the structurally broken interlinks in source DBpedia person snapshot 3.7 to target Freebase[16]. We have conducted this case study by executing following steps: *Step 1* – the interlink dataset[17] that contains links from source to target has been uploaded to the MarkLogic triple store in a distinct graph[18]. *Step 2* – Before repair, we identified the number of structurally broken interlinks from source to target using a different approach than DELTA-R. For this, we used the SUMMR interlink validation template [4]. It identified 704 broken interlinks in source DBpedia. *Step 3* – we have used the detected changes by DELTA-R approach for DBpedia person snapshot 3.6 and 3.7 (second input set) to identify the broken interlinks from source to target. We identified the same no. of broken interlinks using the SPARQL templates available online{LINK}. However, in step 3, we were also able to identify the reason of broken interlinks, i.e. 659, 17, 28 links were broken due to the removed, moved, and renewed resources respectively. *Step 4* – the identified broken interlinks were repaired using the SPARQL templates available online{LINK}. In repair process, the SPARQL templates only deletes the broken interlinks corresponding to the removed resources. For the broken interlinks corresponding to the move and renew resources, the template first deletes the broken interlinks, then adds a new link using the subject URI of the newer resource. The SPARQL template removed the 659 broken interlinks corresponding to the removed resources. Out of 17 broken interlinks corresponding to the moved resources, the SPARQL template identified that 12 repaired interlinks (links using the subject URI of the newer resources) were already there in the interlink dataset. So, the template deleted all the 17 broken interlinks, but added only 5 new interlinks in the interlink dataset. For 28 broken interlinks corresponding to the renewed resources, all the repaired interlinks were already there in the interlink dataset. Hence, 28 broken interlinks were deleted but no new interlink was added. *Step 5* – after repair we have again used the SUMMR interlink validation template to identify the broken interlinks. The template identified 0 broken interlinks.

By means of this case study we demonstrated that DELTA-R approach and DELTA-R change model can support automatic repair of structurally broken

[16] This case study is not about repairing the broken interlinks in target dataset.

[17] http://oldwiki.dbpedia.org/Downloads37#linkstofreebase

[18] http://marklogic.com/semantics/DBpedia/interlinks.nt

interlinks. In future, we will demonstrate the support of DELTA-R approach and DELTA-R change model for other use cases mentioned in the related work as well.

7 CONCLUSION

The paper presents DELTA-R, an approach to detect and classify the changes between two versions of a linked dataset. To represent the detected changes, the paper also presents DELTA-R change model that bridges the gap between resource and triple centric view of changes in linked datasets.

The research question presented in this paper is to what extent we can detect and classify the resource level changes between two versions of a linked dataset.

To answer the research question, an experiment was conducted by applying the DELTA-R approach on DBpedia person snapshots 3.2 (**20,284** resources) and 3.3 (**29,498** resources), and DBpedia person snapshot 3.6 (**296,595** resources) and 3.7 (**790,703** resources). For the former set of snapshots, the approach detected *3819 created*, *239 removed*, *4161 updated*, *124 moved*, and *46 renewed* resources, while on the latter set of snapshots we detected *499590 created*, *5482 removed*, *50380 updated*, *2723 moved*, and *1529 renewed* resources. In the evaluation, we found that the formed hypothesis is supported by the results, as the DELTA-R approach outperforms the state of the art approaches by ~2 - 4 % in terms of accuracy. Also, in comparison to the existing classification of the resource level changes, the more granular classification of changes by DELTA-R approach identified *46* and *1529* additional resources that have changed their representation and may lead to semantically broken interlinks, for former and latter set of datasets respectively.

Finally, we demonstrated the support of DELTA-R approach and change model for interlink maintenance use case. For this, we performed a case study to repair the structurally broken interlinks from DBpedia person snapshot 3.7 to Freebase. In the case study, 704 structurally broken interlinks were repaired. The repaired interlinks were then validated by using SUMMR interlink validation template.

Acknowledgments. This research has received funding from the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded by the European Regional Development Fund.

References

1. Popitsch, N. and Haslhofer, B., 2011. DSNotify–a solution for event detection and link maintenance in dynamic datasets. Web Semantics: Science, Services and Agents on the World Wide Web, 9(3), pp.266-283.

2. Umbrich, J., Decker, S., Hausenblas, M., Polleres, A. and Hogan, A., 2010. Towards dataset dynamics: Change frequency of linked open data sources.
3. Kovilakath, V.P. and Kumar, S.D., 2012, August. Semantic broken link detection using structured tagging scheme. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (pp. 16-20). ACM.
4. Meehan, A., Kontokostas, D., Freudenberg, M., Brennan, R. and O'Sullivan, D., 2016, October. Validating Interlinks Between Linked Data Datasets with the SUMMR Methodology. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 654-672). Springer, Cham.
5. Roussakis, Y., Chrysakis, I., Stefanidis, K., Flouris, G. and Stavrakas, Y., 2015, October. A flexible framework for understanding the dynamics of evolving RDF datasets. In International Semantic Web Conference (pp. 495-512). Springer, Cham.
6. Pourzaferani, M. and Nematbakhsh, M.A., 2013. Repairing broken RDF links in the web of data. International Journal of Web Engineering and Technology, 8(4), pp.395-411.
7. Pernelle, N., Saïs, F., Mercier, D. and Thuraisamy, S., 2016. RDF data evolution: efficient detection and semantic representation of changes. Semantic Systems-SEMANTiCS2016, pp.4-pages.
8. Fernández, J.D., Polleres, A. and Umbrich, J., 2015. Towards Efficient Archiving of Dynamic Linked Open Data. In DIACRON@ ESWC (pp. 34-49).
9. Ward, D.J., Blackwell, A.F. and MacKay, D.J., 2000, November. Dasher—a data entry interface using continuous gestures and language models. In Proceedings of the 13th annual ACM symposium on User interface software and technology (pp. 129-137). ACM.
10. Lee, T., Im, D.H. and Won, J., 2016. Similarity-based Change Detection for RDF in MapReduce. Procedia Computer Science, 91, pp.789-797.
11. Javed, M., Abgaz, Y.M. and Pahl, C., 2014. Layered change log model: bridging between ontology change representation and pattern mining. International Journal of Metadata, Semantics and Ontologies, 9(3), pp.184-192.
12. Patrick, J., Sabbagh, M., Jain, S. and Zheng, H., 2010. Spelling correction in clinical notes with emphasis on first suggestion accuracy. In Proceedings of 2nd Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2010) (pp. 1-8).