PatchR – A Framework for Linked Data Change Requests

Magnus Knuth and Harald Sack
Hasso Plattner Institute for Software Systems Engineering, University of Potsdam, Germany

ABSTRACT

Incorrect or outdated data is a common problem when working with Linked Data in real world applications. Linked Data is distributed over the web and under control of various dataset publishers. It is difficult for data publishers to ensure the quality and timeliness of the data all by themselves, though they might receive individual complaints by data users, who identified incorrect or missing data. Indeed, we see Linked Data consumers equally responsible for the quality of the datasets they use. PatchR provides a vocabulary to report incorrect data and to propose changes to correct them. Based on the PatchR ontology a framework is suggested that allows users to efficiently report and data publishers to handle change requests for their datasets.

Keywords: Linked Data, RDF, data curation, change management, crowd-sourcing

INTRODUCTION

With the continuous growth of Linked Data on the World Wide Web and the increase of web applications that consume Linked Data, the quality of Linked Data resources has become a relevant issue. Recent initiatives, as the Pedantic Web group[1] and DBpedia Data Quality Evaluation Campaign[2] uncovered various defects and flaws in Linked Data resources. Apart from structural defects, semantic flaws and factual mistakes are hard to detect by automatic procedures and require updates on the schema level as well as on the data level.

It is in fact a problem that erroneous data is distributed and reused in various semantic web applications, but it also opens up opportunities for joint efforts such as crowd-sourcing to improve data quality. Indeed, we see Linked Data consumers equally responsible for the quality of the datasets they use within their applications. For example, a semantic web application might offer the possibility of user feedback to signalize facts, which need to be revised. Then, detected errors could be shared with the original data publisher and other users of the dataset. Both would be able to correct the identified defects. While the need of error correction and data cleansing has reached the interest of the Linked Data community there exists no generally accepted method to expose, advertise, and retrieve suitable updates for Linked Data resources. In order to reuse curation efforts and to realize the vision of a collaborative method for error detection and effective exchange of corresponding corrections the following requirements have to be considered:

1. The description of defects and their corresponding fixes for Linked Data re-

[1] https://groups.google.com/group/pedantic-web
[2] http://nl.dbpedia.org:8080/TripleCheckMate/

sources should be facilitated combined with various criteria, e. g. the scope of a fix, provenance information, and the type of defect to select fixes efficiently.
2. The realization of an appropriate workflow that covers guidelines to publish detected errors has to notify the original publishers as well as other users of a particular dataset. To encourage acceptance the application of updates has to be as convenient as possible.
3. Quality improvements for Linked Data resources should also be published as Linked Data to ease their exchange and to make them available for rating, discussions, and reuse.

In this paper we propose an approach that allows users to report Linked Data change requests (patches) within datasets and respective data publishers to effectively process such reports in order to pick up improvement suggestions from the community. The approach consists of the *PatchR* ontology, a framework implementation, and an appropriate workflow.

The arguments for the presented framework start with an overview of related work in the area of Linked Data curation. Next, the overall workflow of requesting Linked Data changes is explained. It allows to expose, rate, and select updates for particular Linked Data resources with a specialized ontology that is described in detail thereafter. Then, the internals of the framework and general usage guidelines are discussed in more detail. The feasibility and technical opportunities of this approach are illustrated exemplary for large knowledge bases, such as DBpedia, where flaws have been detected with the help of human users, in particular with a collaborative data cleansing game (*WhoKnows?*) and a fact ranking tool (*FRanCo*), as well as heuristic data cleansing tools (namely *RDFUnit* and *SDType)*, for single file Linked Data publications, and for ontology evolution scenarios. The created patches are exposed and shared using the herein described ontology. Finally, the conclusion of the paper and an outlook on future work is given.

RELATED WORK

In order to raise quality in Linked Data published on the web multiple efforts aim to assure data consistency. On the one hand tools have been developed to identify erroneous data mainly on syntactic level and on the other hand an increasing number of efforts concentrate on the correction of broken or incomplete data. In this section, first related work on Linked Data validation and error detection is discussed, followed by a discussion of recent efforts on Linked Data correction and enhancement.

Validation

Various work has already focused on checking syntactical and logical data consistency by providing validators for the Semantic Web languages RDF and OWL, e. g. W3C's *RDF Validator*[3], *Vapour*[4], and *OWL Validator*[5]. Poor data quality in Linked Data has been identified, as for instance Hogan et al. analyzed typical errors and set up the *RDF:Alerts Service*[6] that detects syntax and datatype errors to assist Linked Data publishers (Hogan, Harth, Passant, Decker, & Polleres, 2010). Similarly, *LODStats*[7] identifies errors within datasets registered at the CKAN dataset metadata registry (Auer, Demter, Martin, & Lehmann, 2012). However, validators are restricted to syntactical consistency and thus are not able to fulfill the following requirements:

1. Recognize the inconsistent usage of domain or range restricted properties with

[3] http://www.w3.org/RDF/Validator/
[4] http://validator.linkeddata.org/vapour
[5] http://owl.cs.manchester.ac.uk/validator/
[6] http://swse.deri.org/RDFAlerts/
[7] http://stats.lod2.eu/rdfdoc/?errors=1

entities that are not members of the designated class restrictions. Let's consider the following RDF triple:

```
dbp:IKEA dbo:keyPerson
    dbp:Chairman.
```

The RDF triple is syntactically correct, while the range restriction of the property `dbo:keyPerson` implies the entity `dbp:Chairman` to be type of the class `dbo:Person`. But, according to DBpedia, `dbp:Chairman` actually is an untyped entity. Moreover, it is rather a business role and from a user's point of view this might be incorrect because an actual person entity is expected to be a key person of a company.

2. Recognize false facts that do not correspond to the (objective) reality, e. g. the given birthdate of a person can be syntactically correct, when considered as an RDF triple, though it may be factually wrong.
3. Identify missing data. Missing information that is generally included in the dataset for applicable resources can lead to wrong conclusions, as it might as well be omitted for a reason. E. g. the absence of a person's spouse could lead to the assumption the person is unmarried.

Although the existing validators are useful to verify syntactical consistency and correctness, they cannot detect semantic or factual mistakes that are probably evident to a human user. Therefore, an efficient integration of (human) intelligence, i. e. crowdsourcing, is required to detect these kinds of errors. We address this important issue by enabling interoperable exchange of feedback on Linked Data facts. So far, this concept is only sparsely present in the Linked Data community.

Dataset Correction and Enhancement

The DBpedia Data Quality initiative (Zaveri et al., 2013) has evaluated the major Linked Data hub with a semiautomatic process and reports an overall error rate of 11.93%. The process also aims to raise data quality by gradually improving the DBpedia extraction framework and the applied mappings according to the identified data quality problems. The evaluation has been performed on a relatively small sample of 500 entities and exclusively targets the DBpedia and its extraction framework, while the approach presented in this paper is applicable to any Linked Dataset.

Several approaches apply statistical methods to identify shortcomings in Linked Data, such as *SDType* (Paulheim & Bizer, 2013), which is a heuristic method based on the link usage among resources that determines and complements missing type information about entities in DBpedia. It resulted in 1,682,704 new type statements with an achieved overall precision of 99 %. The application of sufficiently expressive ontologies connected to the Linked Data resources allows detecting inconsistent statements also by logical inference. Based on an ontology enriched with assertions deduced from statistical evaluation of DBpedia entities (Töpper, Knuth, & Sack, 2012) have been able to identify inconsistent facts in DBpedia.

Within the LOD2 project (Auer, Lehmann, Ngonga Ngomo, & Zaveri, 2013) tool support for *Evolution and Repair* of Linked Open Data has been developed: *LODRefine* (Verlič, 2012) provides Linked Data extensions and services for OpenRefine[8], a standalone open source desktop application for cleanup and transformation of tabular data to other formats. *LODRefine* enables to triplify, reconcile, and clean tabular data, as well as to export it to RDF. Though the tool is primarily designed for the pre-publishing process, it can also support cleansing Linked Datasets manually.

Harnessing human intelligence for creating semantic content has been studied by

[8] https://github.com/OpenRefine/

Siorpaes and Simperl, who provide a collection of games with a purpose that contribute to the tasks of ontology design, video clip annotation, or ontology matching (Siorpaes & Hepp, 2008; Siorpaes & Simperl, 2010). However, these games generally concentrate on content enrichment rather than on content curation. Another game with a purpose is *WhoKnows?*, a quiz game in the style of 'Who Wants to Be a Millionaire?' published previously by our research group. It is especially designed also to detect errors and shortcomings in DBpedia resources (Waitelonis et al., 2011). Likewise, *RISQ!* is a 'Jeopardy!'-like game that focuses on the evaluation and ranking of Linked Data properties about famous people. Both games are already well accepted but lack a standardized method to publish the obtained curation efforts.

In general, all these achievements are welcome to data providers, more so as data providers might receive regular complaints from the user community because of data quality issues. But since it is very costly to validate and integrate such change requests they are often disregarded and get lost. That also points out the open issue of remote updates in Linked Data management. The W3C's *Read Write Web Community*[9] and *Linked Data Platform*[10] groups work on specifications, infrastructure, and applications for trusted read and write operations. Since it is questionable to what extent publishers will allow direct write access to their data, with our proposed approach the full control over updates remains with the publisher bolstered by the proposals and suggestions of the data consumers.

On the other hand big data providers for Linked Data, as the European digital library *Europeana*, already might collect user feedback but handle curation results only internally (Haslhofer & Isaac, 2011). In the context of Semantic Search Google presents facts about entities from the Knowledge Graph next to search results. These facts are not always correct or up-to-date[11]. Users may report such wrong data by using a feedback button, but are not able to correct it directly or propose a correction.

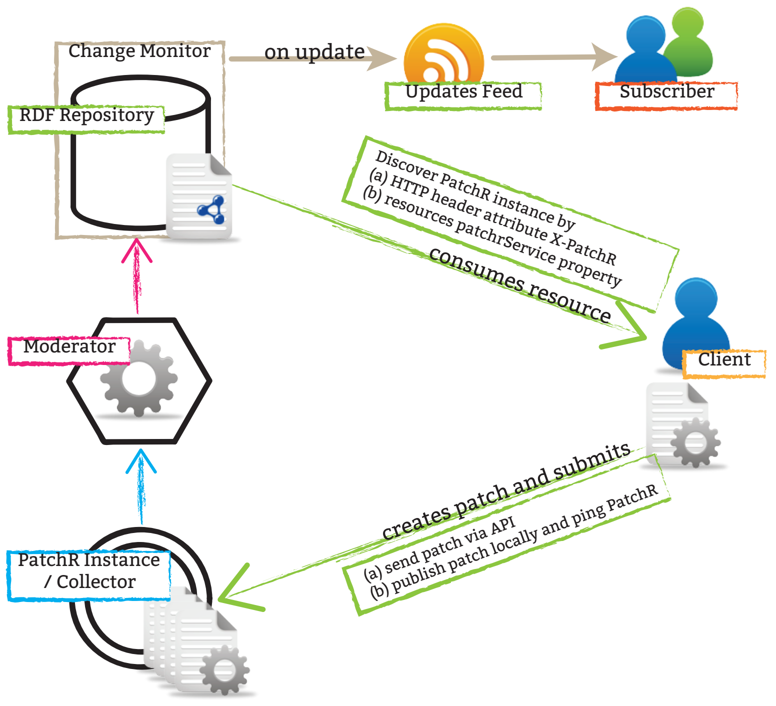
In this paper a new approach is proposed to curate Linked Data in a collaborative way, e. g. flaws in Linked Data resource descriptions that are hard to detect by automatic procedures. With respect to data cleansing of DBpedia resources one could argue that curation efforts should be applied directly to the original sources, i. e. to the online encyclopedia Wikipedia[12]. Indeed, this is the most sustainable practice and data might be correct with the next extraction phase. But, automatic revision of manually curated Wikipedia content does not fit into the guidelines and rules of the Wikipedia community. Therefore, the herein proposed approach goes beyond re-extraction efforts, such as DBpedia Live (Stadler, Martin, Lehmann, & Hellmann, 2010) and can be applied to any Linked Data resource.

WORKFLOW DESCRIPTION

By design, data providers and consumers in the Web of Data are not always the same party. Linked Data explicitly promotes the use of external data resources within own applications. As a result, the datasets are not necessarily under control of the agents who actually employ the dataset, so that they could fix identified inconsistencies by their selves. An overall approach would be to petition the dataset provider to take care of a fix. Though, nowadays it is common practice to set up a private local data store or at least a data cache containing web data as a

[9] http://www.w3.org/community/rww/
[10] http://www.w3.org/2012/ldp/
[11] http://tinyurl.com/outdatedgkg
[12] http://www.wikipedia.org/

Figure 1. Workflow diagram for patch submission and processing


Change Monitor
on update
Updates Feed
Subscriber
RDF Repository
Discover PatchR instance by
(a) HTTP header attribute X-PatchR
(b) resources patchrService property
consumes resource
Client
Moderator
creates patch and submits
(a) send patch via API
(b) publish patch locally and ping PatchR
PatchR Instance
/ Collector

local copy, may it be for reasons concerning performance or data control.

Either way, there is currently no standardized methodology to inform data providers distributing a particular dataset about inconsistencies that have been detected within the data. To tackle this problem, we suggest the *PatchR* vocabulary (c. f. next section) that allows describing various kinds of patch requests including provenance information encoded in RDF.

The proposed PatchR system including its components and their interaction is depicted in the workflow diagram in Fig. 1. Whenever a *Client*, whereby this may be equally a human or software agent, identifies inconsistent facts (RDF triples) within the dataset, he can create a new patch request. The patch request describes the update that has to be performed on the dataset to solve the identified issue. As illustrated with the example in Listing 1 an update consists out of a set of RDF triples to add and/or a set of RDF triples to delete in the particular dataset. The patch request needs to be submitted to the *PatchR Instance* belonging to the dataset. To achieve this, either a PatchR side API can be applied to send the patch request directly, or the client publishes the patch request in an own repository and announces this publication via semantic pingback RPC service (Tramp, Frischmuth, Ermilov, & Auer, 2010). The decision on the execution of particular patch requests is left to the publisher's *Moderator* instance based on individual rules.

The framework also encourages giving access to collected patch requests to data consumers in order to let them vote for or against proposed updates. Furthermore, modifications in the dataset should be propagated via an *Update Feed* that allows subscribers to update local copies of the dataset or to invalidate their caches.

Listing 1. An example patch request

```
@prefix :     <http://example.org/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix pat:  <http://purl.org/hpi/patchr#> .
@prefix guo:  <http://webr3.org/owl/guo#> .
@prefix prov: <http://purl.org/net/provenance/ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix dbp:  <http://dbpedia.org/resource/> .
@prefix dbo:  <http://dbpedia.org/ontology/> .

:Patch_15 a pat:Patch ;
   pat:appliesTo <http://dbpedia.org/void.ttl#DBpedia_3.5> ;
   pat:status pat:Open ;
   pat:update [
      a guo:UpdateInstruction ;
      guo:target_graph <http://dbpedia.org/> ;
      guo:target_subject dbp:Oregon ;
      guo:delete [
         dbo:language dbp:De_jure ] ;
      guo:insert [
         dbo:language dbp:English_language ]
      ] ;
   prov:wasGeneratedBy [
      a prov:Activity ;
      pat:confidence "0.5"^^xsd:decimal ;
      prov:wasAssociatedWith :WhoKnows ;
      prov:actedOnBehalfOf :WhoKnows#Player_25 ;
      prov:performedAt "..."^^xsd:dateTime ] .
```

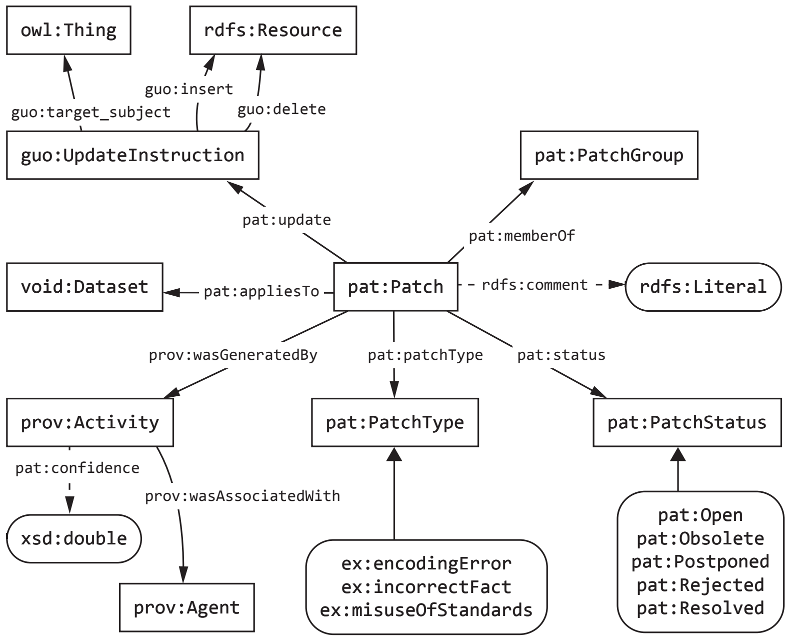
DESCRIPTION OF THE PATCH REQUEST ONTOLOGY

The Patch Request Ontology (`pat`)[13] (Knuth, Hercher, & Sack, 2012), subsequently referred to as *PatchR*, allows expressing change requests within a Linked Data dataset that is under control of an external publisher. Since *PatchR* wraps the `guo:UpdateInstruction` concept adopted from the *Graph Update Ontology* (`guo`)[14], change requests are always entity-centric, i. e. within a patch request `pat:Patch` a `foaf:Agent` may demand the insertion, deletion, or substitution of (a sub-graph of) RDF triples related to a single entity. The main constituent of the *PatchR* ontology is a patch request (`pat:Patch`). Each patch is endorsed by provenance information provided by the Provenance Vocabulary Core Ontology (`prov`)[15] and a `void:Dataset` it applies to. Furthermore, a patch can be classified using the `pat:patchType` property to allow efficient retrieval of common patches. These patch types may refer to commonly observed errors, e. g. encoding problems or factual errors. There might be patch tax-

[13] cf. http://purl.org/hpi/patchr
[14] cf. http://webr3.org/owl/guo

[15] cf. http://trdf.sourceforge.net/provenance

Figure 2. Overview on the patch request ontology



onomies from different applications that define the reason for a patch on their own.

Individual patches can be bundled into a `pat:PatchGroup`, e. g. patches of a particular service that apply to a common problem or have relevance only for specialized domains or regions. Fig. 2 provides an overview on the main concepts of the *PatchR* ontology, which are described in Table 1.

Dedicated error detection algorithms often base on statistics and are to some extent uncertain. To express this level of certainty in a patch, the confidence can be stated as a numeric value in the range of [−1, 1], whereas higher values indicate a higher certainty and negative values attribute criticism towards this patch request. This confidence value is bound to the agent's provenance information for the patch. In case multiple agents propose the same patch the total confidence for the patch raises. By collecting the confidences of multiple agents a shared commitment can be reached, expressed as a numerical value that sums agreement and subtracts disagreement on a patch request. Therefore, confidence values of multiple agents ($conf_{p|a}$ and $conf_{p|b}$) of the same patch *p* can be combined by an associative, commutative operation. We proposed a uniformly continuous operation ⊕ in (0, 1], i. e. for combining positive confidences, and the inclusion of trust values towards individual agents in (Knuth & Sack, 2014):

$$\begin{aligned} conf_{p|a,b} &= conf_{p|a} \oplus conf_{p|b} \\ &= 1-((1-conf_{p|a})\cdot(1-conf_{p|b})) \\ &= conf_{p|a} + conf_{p|b} - (conf_{p|a}\cdot conf_{p|b}) \end{aligned}$$

A central aspect considered by the ontology is voting for or against patch requests. Therefore, votes of advocates and criticizers can be linked to an existing patch as additional `prov:Activity`s with appropriate confidence values: any positive confidence will raise the total confidence, any negative confidence will reduce the total patch confidence.

PATCHR FRAMEWORK

The proposed framework comprises several components on consumer (client) and pub-

Table 0. Description of properties in the patch request ontology

Property	Description
`update`	Refers to a `guo:UpdateInstruction`. There must be exactly one `guo:UpdateInstruction` per patch.
`memberOf`	Assignment of a patch to a `pat:PatchGroup`.
`appliesTo`	Refers to a `void:Dataset` to allow convenient selection of patches per dataset.
`patchType`	Refers to a classification of a patch. A patch can have multiple types.
`confidence`	A confidence assigned by the creator of the patch, e. g. in case heuristic methods identified an inconsistency. This confidence must be expressed in the range of [−1, 1], whereas a high value means higher confidence and a value of 1 signifies absolute certainty. Negative values indicate criticism towards this patch.
`status`	The status of the patch, might be one of `pat:Open`, `pat:Resolved`, `pat:Obsolete`, `pat:Postponed`, or `pat:Rejected`.

lisher (server) side. In the following these components and their behavior is described in more detail.

Client behavior

A *PatchR Client* consumes Linked Data, i. e. resource descriptions, from a publisher's repository directly or via a local copy or cache. Any Linked Data consuming application that includes some sort of data validation or inconsistency detection may serve as a PatchR client. That could be for example a simple interface to display information from the data repository to a human user, which provides a reporting functionality, or also a dedicated data cleansing algorithm. Once an inconsistency is detected by the client, it creates a patch request to delete, add, or alter RDF triples that relate to a particular subject resource using the PatchR vocabulary as described in the previous section. For the creation and management of patch requests a Java API[16] is provided.

[16] Available on GitHub: https://github.com/mgns/PatchR

To announce the patch request the client needs to discover the URL of the *PatchR instance* responsible for the respective resource. To allow the client the auto-discovery of the collector instance an HTTP response header attribute `X-PatchR` should be set when sending a resource description that holds the respective URL of the PatchR server. Alternatively and for a better integration with Linked Data principles, an OWL property `pat:patchrService`, can be used to link the PatchR server directly to the resource or dataset description as shown in Listing 2. The advantages of using the property would be to cache this information locally and the possibility to assign individual PatchR server instances to Hash URIs as well, while the header attribute approach only requires an HTTP header request instead of retrieving and parsing the full resource description.

Since patch requests are RDF resources a method to transmit RDF data is required. To achieve this and to submit a patch request two options are considered suitable: Either, clients may call a dedicated API provided by the *PatchR Instance* to submit patches.

Listing 2. Usage example of the patchrService property

```
<http://magnus.13mm.de/> pat:patchrService
      <http://patchr.s16a.org/> .
```

Therefore, a Linked Data serialization and messaging format such as the JSON-LD standard[17] can be applied. The API also will need an identification mechanism to ensure the authorship, e. g. by API key or WebID. Alternatively, the patch request can also be published at a client side web server. In this case the collector needs to be informed about the creation, which can be carried out through the semantic pingback mechanism (Tramp et al., 2010). Hereby, the client calls an RPC method on the *Collector* side having the URL of the patch as an argument. This enables the PatchR service to retrieve the patch from the client's repository.

Server behavior

The *Collector* receives and validates patches from the clients, and stores them in a local repository that should allow open access to retrieve patches.

For further processing of the collected data, the *Moderator* decides about the application of submitted patches to the dataset individually. This module needs to employ guidelines custom-made for the dataset owner. Such guidelines might be represented by a simple majority vote, but might also consider the submitter's reputation or given trust, e. g. obtained by previous submissions of high quality patch requests. A final decision about the application of a patch should be recorded using the `pat:status` property (set from "Open" to "Obsolete", "Rejected", "Postponed", or "Resolved").

To keep dataset consumers informed about updates to the original dataset a *Change Monitor* should report changes, e. g. as an RSS feed or by proactive notification.

[17] http://www.w3.org/TR/json-ld/

The granularity and range of the reports could be arbitrary, e. g. including the whole dataset, individual resources, or sets of resources.

USE CASES FOR PATCHR

In this section, the application of the *PatchR* ontology is demonstrated in several scenarios that cover crowd-based and heuristic agents for the large DBpedia knowledge base, a submission mechanism for single resource Linked Data publications, and an ontology evolution scenario, which may facilitate user involvement in the development process.

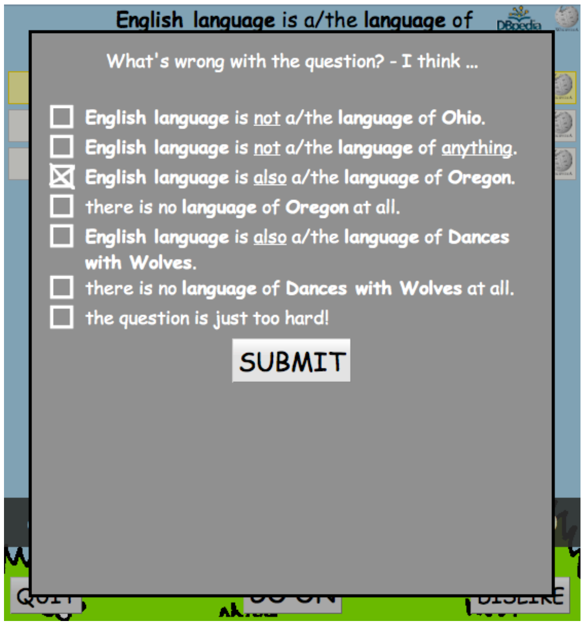
DBpedia

DBpedia is an openly available, multi-domain and multilingual RDF dataset extracted from Wikipedia content and a major source for structured knowledge on the Web (Auer et al., 2007). The latest release (DBpedia 2014) consists of over 880 million RDF triples[18] describing 4.58 million entities. DBpedia is heavily interlinked with other datasets and plays a central role in the Linked Open Data cloud. It is therefore a suitable data source for integration in cross-domain Linked Data applications, such as document annotation, faceted search, location-based information services, information extraction, and natural language processing services. Nevertheless, it became obvious that DBpedia lacks sufficient data quality for a range of applications (Zaveri et al., 2013).

[18] Number of triples available via the DBpedia SPARQL endpoint and published as Linked Data, more triples are available as RDF dump files.

Patching the DBpedia knowledge base can be regarded as a special case, since this data is based on Wikipedia articles that are curated manually by the Wikipedia community. Identified problems should sustainably be fixed in the respective Wikipedia article source page or the DBpedia Extraction Framework[19] or Mappings Wiki[20], so they won't occur in future DBpedia releases. Since DBpedia is freshly extracted and released approximately once a year, change requests should apply for a particular version. With every new release change requests might become obsolete in case triples requested for deletion have vanished or triples requested for insertion have been introduced to the new version. Because of that, one could argue that patch requests for prior DBpedia versions become useless and the patching process has to start from it's beginning, but since DBpedia resources are in general not substantially changing over versions and resource URIs are more or less stable, we see that patches need to be checked for obsolescence when migrating to a newer version but remain valid as the inconsistent facts persist. Due to it's sheer size it is unlikely to fix all errors in DBpedia by patches. Nevertheless, patches can give useful hints to identify structural problems in the extraction process.

Figure 3. Screenshot of WhoKnows?' refinement panel



English language is a/the language of

What's wrong with the question? - I think ...

☐ English language is not a/the language of Ohio.
☐ English language is not a/the language of anything.
☒ English language is also a/the language of Oregon.
☐ there is no language of Oregon at all.
☐ English language is also a/the language of Dances with Wolves.
☐ there is no language of Dances with Wolves at all.
☐ the question is just too hard!

Crowd-based Agents. As presented in (Knuth et al., 2012), the *WhoKnows?* game (Waitelonis et al., 2011) has been extended with support for the *PatchR* ontology[21]. The game's principle is to present multiple-choice questions to the user that have been directly generated out of facts from DBpedia RDF triples. The player scores points by providing correct answers within a limited amount of time and loses lives whenever he gives a wrong answer or no answer at all. Due to inconsistent or incorrect facts in the knowledge base, questions or expected answers may appear defective to the player. In such a case the player can report the question by clicking a "Dislike" button. Thereon, the player is asked to specify the particular fact, which he thinks to be incorrect by selecting it from a given set of constructed potential inconsistencies. For simplicity, the potential inconsistent RDF triples are presented to the user as natural language sentences. Fig. 3 shows a screenshot of this refinement panel. Sentences of the form *'Object is not a property of subject.'* indicate a wrong fact in the dataset, while sentences of the form *'Object is also a property of subject.'* indicate a missing fact. From this user vote the system generates a patch request for either

- deleting one or several triples or
- inserting one or several triples in the underlying knowledge base.

[19] https://github.com/dbpedia/extraction-framework

[20] http://mappings.dbpedia.org/

[21] Play *WhoKnows?* at http://tinyurl.com/whoknowsgame

Listing 3. Patch requests from DBpedia 3.5.1 lasting in 2014 (Open)

```
## Undeleted
dbp:Aldiborontiphoskyphorniostikos dbo:country dbp:London .
dbp:Arbors_Records dbo:distributingCompany dbp:United_States .
dbp:Diedrich_Coffee dbo:keyPerson dbp:President .
dbp:Jordanus_de_Nemore dbo:nationality dbp:Europe .
dbp:Monster_Magnet dbo:hometown dbp:United_States .

## Not Inserted
dbp:George_F._Kennan dbo:birthPlace dbp:United_States .
dbp:Oregon dbo:language dbp:English_language .
dbp:Turkey dbo:governmentType dbp:Federalism .
dbp:Volvo dbo:product dbp:Automobile .
dbp:YouTube dbo:language dbp:English_language .
```

Listing 4. Removed/inserted triples in DBpedia 2014 as requested (Obsolete)

```
## Removed
dbp:Liz_Carroll dbo:hometown dbp:United_States .
dbp:Quest_Software dbo:keyPerson dbp:President .
dbp:Singapore dbo:language dbp:English_alphabet .
dbp:United_States_Senate dbo:meetingCity dbp:United_States .
dbp:William_Douglas_Crowder dbo:country dbp:United_States_Navy .

## Inserted
dbp:Combined_Arms_Research_Library dbo:country dbp:United_States .
dbp:Life_Sentence_Records dbo:country dbp:United_States .
dbp:Louisiana dbo:language dbp:English_language .
dbp:Texas dbo:language dbp:English_language .
dbp:United_Kingdom dbo:regionalLanguage dbp:Cornish_language .
```

WhoKnows? which is originally based on DBpedia 3.5.1 delivered 4,819 unique patch requests (5,605 including multiple reports of the same change request). These patch requests have been made publicly available by a simple user interface[22] including respective links to the DBpedia resource and the Wikipedia article where the involved facts originated.

It can be assumed that due to these reports users have resolved only a minority of the reported bugs in Wikipedia articles. Nevertheless, in the current version of DBpedia (DBpedia 2014) only 2,657 patches (55.1 %) remain open, while 2,162 patches turned obsolete due to changes to Wikipedia or the DBpedia extraction framework. Listings 3 and 4 show a number of open and obsolete change requests. This shows that the players of the *WhoKnows?* game proposed reasonable changes.

FRanCo (Fact Ranking Evaluation)[23] is a crowd-sourcing approach which collects rankings of RDF triples for a sample of 541 individual DBpedia resources (DBpedia 2014) aiming to create a ground-truth for fact ranking evaluation (Bobić, Waitelonis,

[22] http://tinyurl.com/patchrui

[23] http://s16a.org/fr/

& Sack, 2015). By doing that, it also offers the possibility to highlight facts as faulty or nonsense statements. Reported facts include `dbp:Joma dbo:product dbp:Tennis`, `dbp:Barbera dbo:wineRegion dbp:California_wine`, and `dbp:Peugeot_504 dbo:assembly dbp:River_Thames`.

In order to improve the data quality of the DBpedia dataset, these triples should be deleted from the knowledge base. So far, we collected 2,494 corresponding change requests from 203 unique users.

Heuristic Linked Data Cleansing Approaches. So far, the quality of extracted triples in DBpedia is primarily improved by post-processing techniques. One of these approaches is *SDType* (Paulheim & Bizer, 2013), which provides missing type information for resources without a known given type to the DBpedia dataset. An entity's class membership is deduced based on statistics about the usage of properties with entities of known type. Due to the statistical character of this approach, the results are somewhat vague and in order to achieve a high precision lower ranked type mappings are not accepted for application to the dataset. As a result, the additional types still contain incorrect mappings while on the other hand valid mappings have been excluded due to their potential vagueness. We suggest to report lower ranked type mappings as patch requests.

RDFUnit (a.k.a. "Databugger") (Kontokostas et al., 2014) allows to unit test RDF(S) datasets. Therefore, tests are generated in the form of SPARQL queries according to predefined patterns based on ontological constraints contained in the vocabularies used in the dataset. The introduction of additional restrictions to the applied schema leads to a higher test coverage. Additionally, custom tests can be defined for the dataset. The tests are generally applicable and are validated by executing the queries on the dataset. In case of faulty data a non-empty result set is returned, which means the test failed. Likewise, *Inconsistency Checker* (Töpper et al., 2012) detects logical inconsistencies in DBpedia using a reasoner-based approach on a previously enriched ontology model with strict type constraints and trained class disjointness. While *RDFUnit*, due to the nature of SPARQL, expects all statements explicitly stated in the dataset, the reasoner-based approach assumes an open world and detects logical conflicts. It may produce inconsistency explanations that include deduced facts as well.

Originally, these tests only indicate erroneous RDF triples in the dataset but do not directly provide solutions to correct them. Though for some patterns generic solutions can easily be identified, that may be either the insertion of missing triples or the deletion of triples that cause the test failure. E. g., the *RDFUnit* RDFSRANGE pattern instantiated with the property `dbo:nationality` would identify a test failure for the triple `dbp:Jordanus_de_Nemore dbo:nationality dbp:Europe` because `dbp:Europe` is not of class `dbo:Country`. This failure can be fixed either by removing the causing triple or adding the type information to the subject. Either solution would solve the issue, but it is not trivial to mechanically decide which solution is better suitable.

Static File Linked Data Publications

Static RDF files and exporters are an easy way to publish Linked Data on the Web. It is often used for low-volume Linked Data publications, such as personal FOAF or SIOC profiles. Such static files are often exported by tools like the FOAF-a-Matic[24] or plat-

[24] http://www.ldodds.com/foaf/foaf-a-matic

forms such as Wordpress and Drupal, and afterwards manually curated.

Since curating such static files is laborious, they often fall into oblivion, and contain outdated information or broken links to external resources. Providing a PatchR instance as a service for such micro-publishers would be beneficial for consumers of this data and the data owner as well. As corrections are reported to the PatchR service, the user gets notified via electronic mail or such and decides on accepting or rejecting the suggested change.

Additionally, the platform could provide a file edit script that finally accomplishes the change, to simplify the maintenance of the publication just as the erstwhile creation.

Ontology Evolution

As OWL ontologies can be represented as RDF their selves, they are equally suitable as RDF datasets for patching. Especially during the development phase of a vocabulary, it can be fruitful to collect various opinions on the technical design from potential future end users. Instead of discussing design changes in informal communication the collaborators could raise change requests towards the vocabulary and vote for or against suggestions of other members.

In such a setting, the voting mechanism may become more relevant and democratic decisions can be found more easily.

CONCLUSION AND OUTLOOK

In this paper we have presented *PatchR*, an approach for involvement of the actual Linked Data consumers into the processes of Linked Data quality improvement, Linked Data evolution, and Linked Data maintenance. We described how the *PatchR* ontology can be used to describe change requests, and an appropriate workflow.

The feasibility of this approach has been illustrated with multiple use cases that rely on human actors as well as algorithmic data curation systems on top of DBpedia. However, this concept is not limited to DBpedia, but can be applied to any dataset that obeys Linked Data principles, such as other knowledge bases, small-scale Linked Data publications, and web ontologies.

Therefore, the presented approach can be of concern for various data providers that are interested in data curation issues. In general each aggregator of Linked Data can help to leverage structural and factual quality of the semantic web. The following steps are necessary for active participation:

1. Identify potential flaws in original or aggregated data.
2. Implement facilities to gather user feedback.
3. Serialize identified flaws and corresponding updates using the *PatchR* ontology.
4. Publish these patches within an appropriate repository that can be publicly accessed.

In regards to managing distributed information on patches we suggest a rather centralized setting, where major dataset providers rely on dedicated patch repositories to obtain patches for their particular dataset. Further auxiliary tasks for effective synchronization of patches such as further standardization and management of trust are not covered in this publication and subject of future research. By providing provenance information for each patch request, the data user may decide for himself whether to apply the proposed patch or – whenever in doubt – to work with the original data or apply a competing but more trustworthy patch.

The current implementation of the framework is available on GitHub[25]. It provides a Java API for patch creation and basic repository management. On top of that client side interfaces for creating and submitting

[25] https://github.com/mgns/PatchR

patch requests to a patch repository has been implemented. A web interface allows the publication of the repository's content and browsing the patch requests. Furthermore, it provides a REST-API to give access to the content and to collect patch requests from external applications. Further work will include the implementation of more advanced trust and access control mechanisms. Features like rating, feedback, and reputation management are mandatory to provide appropriate incentives in the long run. A pingback mechanism might be valuable to inform data providers about recently created patch requests concerning their datasets.

It is planned to publicly provide the interface to users of DBpedia and DBpedia Live. Because it is not intentional to apply changes directly to the DBpedia datasets, we rather purpose a wrapper that allows dereferencing individual DBpedia resources with patches applied on the fly.

To enable the application of distributed and publicly maintained Linked Data resources in professional applications, sufficient data quality standards have to be ensured. The proposed *PatchR* framework is one step towards the support and maintenance of Linked Data quality standards throughout the data lifecycle, which is essential for adaption in a professional environment.

REFERENCES

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: a nucleus for a web of open data. In *The Semantic Web, ISWC 2007* (Vol. 4825, pp. 722–735). Springer.

Auer, S., Demter, J., Martin, M., & Lehmann, J. (2012, October). LODStats – an extensible framework for high-performance dataset analytics. In *Proceedings of 18th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2012* (pp. 353–362). Galway, Ireland: Springer.

Auer, S., Lehmann, J., Ngonga Ngomo, A.-C., & Zaveri, A. (2013). Introduction to linked data and its lifecycle on the web. In *Proceedings of the 9th International Conference on Reasoning Web: Semantic Technologies for Intelligent Data Access* (pp. 1– 90). Berlin, Heidelberg: Springer.

Bobić, T., Waitelonis, J., & Sack, H. (2015). FRanCo – a ground truth corpus for fact ranking evaluation. In *Proceedings of the 1st International Workshop on Summarizing and Presenting Entities and Ontologies, SumPre 2015.* (submitted)

Haslhofer, B., & Isaac, A. (2011). data.europeana.eu – The Europeana Linked Open Data Pilot. In *Proceedings of the International Conference on Dublin Core and Metadata Applications.*

Hogan, A., Harth, A., Passant, A., Decker, S., & Polleres, A. (2010). Weaving the pedantic web. In *Linked Data on the Web Workshop, LDOW 2010, co-located with the 19th International World Wide Web Conference, WWW 2010* (Vol. 628, pp. 30–34). CEUR-WS.

Knuth, M., Hercher, J., & Sack, H. (2012, April). Collaboratively patching linked data. In *Proceedings of 2nd International Workshop on Usage Analysis and the Web of Data, USEWOD 2012, co-located with the 21st International World Wide Web Conference, WWW 2012,* Lyon, France. archix.org.

Knuth, M., & Sack, H. (2014, May). Data cleansing consolidation with PatchR. In *Proceedings of the 11th Extended Semantic Web Conference, ESWC 2014, Satellite Events.* Springer.

Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., & Zaveri, A. J. (2014). Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web, WWW 2014* (pp. 747–758). ACM.

Paulheim, H., & Bizer, C. (2013). Type inference on noisy RDF data. In *The Semantic Web, ISWC 2013* (Vol. 8218, pp. 510–525). Springer.

Siorpaes, K., & Hepp, M. (2008). OntoGame: Weaving the semantic web by online games. In *Proceedings of the 5th European Semantic Web Conference, ESWC 2008* (pp. 751-766). Springer.

Siorpaes, K., & Simperl, E. (2010). Human intelligence in the process of semantic content creation. *World Wide Web* (Vol. 13(1–2), pp. 33–59). Springer.

Stadler, C., Martin, M., Lehmann, J., & Hellmann, S. (2010, May). Update strategies for DBpedia Live. In *6th Workshop on Scripting and Development for the Semantic Web, SFSW 2010, co-located with 7th Extended Semantic Web Conference, ESWC 2010* (Vol. 699). CEUR-WS.

Töpper, G., Knuth, M., & Sack, H. (2012, September). DBpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems, i-Semantics 2012* (pp. 33–40). ACM.

Tramp, S., Frischmuth, P., Ermilov, T., & Auer, S. (2010). Weaving a social data web with semantic pingback. In *Knowledge Engineering and Management by the Masses* (Vol. 6317, p. 135–149). Springer.

Verlič, M. (2012, September). LODGrefine – LOD-enabled Google Refine in action. *Proceedings of the 8th International Conference on Semantic Systems, i-Semantics 2012 (Posters & Demos)* (pp. 31–37). ACM.

Waitelonis, J., Ludwig, N., Knuth, M., & Sack, H. (2011). Who-Knows? – Evaluating linked data heuristics with a quiz that cleans up DBpedia. *International Journal of Interactive Technology and Smart Education, ITSE* (Vol. 8(3), pp. 236–248). Emerald.

Zaveri, A., Kontokostas, D., Sherif, M. A., Bühmann, L., Morsay, M., Auer, S., & Lehmann, J. (2013, September). User-driven quality evaluation of DBpedia. In *Proceedings of the 9th International Conference on Semantic Systems, i-Semantics 2013* (pp. 97–104). ACM.