Semantic Web 0 (0) 1–0
IOS Press

Knowledge Base Quality Assessment Using Temporal Analysis

Mohammad Rashid [a,*], Marco Torchiano [a], Giuseppe Rizzo [b], Nandana Mihindukulasooriya [c] and Oscar Corcho [c]

[a] *Politecnico di Torino, Italy*
E-mail: mohammad.rashid@polito.it, marco.torchiano@polito.it
[b] *Instituto Superiore Mario Boella*
E-mail: giuseppe.rizzo@ismb.it
[c] *Universidad Politecnica de Madrid, Spain*
E-mail: nmihindu@fi.upm.es, ocorcho@fi.upm.es

Abstract. Knowledge bases are nowadays essential components for any task that requires automation with some degrees of intelligence. The quality of such knowledge bases can drastically affect the decisions being taken by any algorithm, thus, for instance, affecting the classification of an email or the final policy maker choice. Establishing checks to ensure a high-level quality of the knowledge base content (i.e. data instances, relations, and classes) is at utmost importance. In this paper, we present a novel knowledge base quality assessment approach that relies on temporal analysis. The proposed approach compares consecutive knowledge base releases to compute quality measures that allow detecting quality issues. In particular, we considered four quality characteristics: Persistency, Historical Persistency, Consistency, and Completeness. The approach has been assessed both quantitatively and qualitatively on a series of releases from two knowledge bases, eleven releases of DBpedia and eight releases of 3cixty Nice. In particular, a prototype has been implemented using the R statistical platform. The capability of Persistency and Consistency characteristics to detect quality issues varies significantly between the two case studies. The Completeness characteristic is extremely effective and was able to achieve 95% precision in error detection. The proposed approach delivered good performances. The measures are based on simple operations that make the solution both flexible and scalable.

Keywords: Knowledge Base, Linked Data, Quality Assessment, Quality Issues, Temporal Analysis

1. Introduction

The Linked Data approach consists in exposing and connecting data from different sources on the Web by the means of semantic web technologies. Tim Berners-Lee [1] refers to linked open data as a distributed model for the Semantic Web that allows any data provider to publish its data publicly, in a machine readable format, and to meaningfully link them with other information sources over the Web. This is leading to the creation of Linked Open Data (LOD) cloud hosting several Knowledge Bases (KBs) making available billions of RDF triples from different domains such as Geography, Government, Life Sciences, Media, Publication, Social Networking, User generated [2].

The knowledge bases are rapidly evolving since both data instances and ontologies are updated, extended, revised, and refactored [13] covering more and more topical domains. In particular, KB instances evolve over time given that new resources are added, old resources are removed, and links to resources are updated or deleted. For instance, DBpedia [2] has been available from the very beginning of the LOD movement and released various versions periodically. Along

*Corresponding author. E-mail: mohammad.rashid@polito.it

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

[2] http://lod-cloud.net

with each release, DBpedia proposed changes at both instance and schema level over time. In particular, the schema level changes involved classes, properties, axioms, and mappings to other ontologies [25]. Usually, instance-level changes include resources typing, property values, or identify links between resources.

KB evolution is important for wide range of applications: effective caching, link maintenance, and versioning [18]. For the KB development and maintenance, *data quality* remains a critical aspect to obtain trust by the users. Data quality, in general, relates to the perception of the "fitness for use" in a given context [35]. The knowledge bases published in the LOD cloud are diverse because of different formats, structures, and vocabularies. Assessing data quality in KB development is a challenge as the data derive from many autonomous, evolving, and increasingly large data sources. For instance, various data integration tasks such as synchronization, data linking, or fusion may be directly impacted by KB evolution [28]. Such transformation may lead to incomplete or incorrect results in the KB generation. Therefore, it is important to understand the KB evolution thus allowing detecting changes and enabling to identify possible data quality issues.

In particular, KBs are dynamic by nature and constantly growing [18], [27]. Using the evolution history we can detect KB changes over time. Based on the detected changes, we aim to analyze and benchmark quality issues in any knowledge base. The main premise that has guided our investigation is: *temporal analysis of the changes observed across different KB releases can lead to detect and measure quality issues.*

In this paper, we address the challenges of quality measurements using temporal analysis. We thus propose a KB quality assessment approach using quality measures that are computed using a temporal analysis. We further divide this research question into three sub-questions:

RQ1: *Which temporal measures can be used to assess KB quality characteristics?*
We propose time-related metrics that can be used to detect quality issues and address quality characteristics.

RQ2: *Which quality assessment approach can be defined on top of the the temporal quality characteristics?*
We propose an approach that profiles different releases of the same KB and it measures automatically the quality of the data.

RQ3: *How to validate the quality measures of a given KB?*
We propose both quantitative and qualitative experimental analysis on two KBs, namely DBpedia and 3cixty.

A large number of quality characteristics followed by measures for computing quality is available in the literature [38]. However to identify uniform quality dimensions, in our approach we followed the standard guidelines of data quality standard ISO/IEC 25024 [17] and W3C DQV [16] to identify temporal-based quality measures. We thus defined four temporal-based quality characteristics based on change detection.

We use basic statistics (i.e. counts, and diffs) over extracted triples from various KB releases to measure quality characteristics. More specifically, we compute the frequency of predicate and the frequency of entities of a given resource type, and we compare the frequencies with the ones observed in previous releases. We validate our approach based on quantitative and qualitative approaches. For the qualitative approach we manually check precision by examining the results from quantitative analysis.

The main contributions of this work are:

- We propose four quality characteristics based on change detection of a KB over various releases;
- We present an automatic method for benchmarking quality issues using change detection over the different KB releases;
- We experiment our approach on two KBs: DBpedia (encyclopedic data) and 3cixty (contextual tourist and cultural data).

This paper is organized as follows:

- In Section 2, we present the related work focusing on change detection and quality measurement of linked data;
- In Section 3, we introduce how quality issues can be inferred with a temporal analysis;
- In Section 4, we describe our approach that relies on temporal analysis and generates automatic measures concerning the quality of a KB;
- In Section 5, we present our empirical evaluation conducted on two different KBs, namely DBpedia and 3cixty Nice KB;
- In Section 6, we further discuss on initial hypotheses and insights;
- In Section 7, we conclude by summarizing the findings of our experimental study and provide answers to the formulated research questions.

2. Related Work

The research activities related to our approach fall into two main research areas: *(i)* Change Detection in Linked Datasets and *(ii)* Linked Open Data Quality Assessment.

2.1. Change Detection in Linked Data

Various research endeavours focus on change detection in linked datasets on various issues. Umbrich *et al.* [36] present a comparative analysis on LOD datasets change and which sensible measures there are to accommodate dataset dynamics. Pernelle *et al.* [28] present an approach that detects and semantically represents data changes in RDF datasets.

Klein *et al.* [19] analyze ontology versioning in the context of the Web. They look at the characteristics of the release relation between ontologies and at the identification of online ontologies. Then they describe a web-based system to help users to manage changes in ontologies.

Ruan *et al.* [32] categorized quality assessment requirements into three layers: understanding the characteristics of data sets, comparing groups of data sets, and selecting data sets according to user-denied scenarios. Based on this, they designed a tool – KBMetrics – to incorporate the above quality assessment purposes. In the tool, they focused to incorporate different kinds of metrics to characterize a data set, but it has also adopted ontology alignment mechanisms for comparison purposes.

Käfer *et al.* [18] present a design and results of the Dynamic Linked Data Observatory. They setup a long-term experiment to monitor the two-hop neighbourhood of a core set of eighty thousand diverse Linked Data documents on a weekly basis. They look at the estimated lifespan of the core documents, how often it go on-line or offline, how often it change as well as they further investigate domain-level trends. They explored the RDF content of the core documents across the weekly snapshots, examining the elements (i.e., triples, subjects, predicates, objects, classes) that are most frequently added or removed. In particular they investigate at how the links between dereferenceable documents evolves over time in the two-hop neighbourhood.

Gottron and Gottron [13] analyse the sensitivity of twelve prototypical Linked Data index models towards evolving data. They addressed the impact of evolving Linked Data on the accuracy of index models in providing reliable density estimations.

Papavasileiou *et al.* [27] address change management for RDF(S) data maintained by large communities, such as scientists, librarians, who act as curators to ensure high quality of data. Such curated KBs are constantly evolving for various reasons, such as the inclusion of new experimental evidence or observations, or the correction of erroneous conceptualizations. Managing such changes poses several research problems, including the problem of detecting the changes (delta) among versions of the same KB developed and maintained by different groups of curators, a crucial task for assisting them in understanding the involved changes. They addressed this problem by proposing a change language which allows the formulation of concise and intuitive deltas.

2.2. Linked Data Quality Assessment

The majority of the related work on Linked Data quality assessment are focused on defining metrics to quantify the quality of data according to various quality dimensions and designing framework to provide tool support for computing such metrics.

Most early work on Linked Data quality were related to data trust. Gil and Arts [11] focus their work on the concept of reputation (trust) of web resources. The main sources of trust assessment according to the authors are direct experience and user opinions, which are expressed through reliability (based on credentials and performance of the resources) and credibility (users view of the truthfulness of information). The trust is represented with a web of trust, where nodes represent entities and edges are trust metrics that one entity has towards the other.

Gamble and Goble [10] also focus on evaluating trust of Linked Data datasets. Their approach is based on decision networks that allow modeling relationships between different variables based on probabilistic models. Furthermore, they discuss several dimensions of data quality: 1. *Quality dimension*, which is assessed against some quality standard and which intends to provide specific measures of quality; 2. *Trust dimension*, which is assessed independently of any standard and is intended to asses the reputation; 3. *Utility dimension*, which intends to assess whether the data fit the purpose and satisfy the needs of users.

Shekarpour and Katebi [34] focus on assessment of trust of a data source. They first discuss several models of trust (centralized model, distributed model, global

model and local model), and then develop a model for assessment of trust of a data source based on *(a)* propagation of trust assessment from data source to triples, and then *(b)* aggregation of all triple assessments.

Golbeck and Mannes [12] focus on trust in networks and their approach is based on the interchange of trust, provenance, and annotations. They have developed an algorithm for inferring trust and for computing personal recommendations using the provenance of already defined trust annotations. Furthermore, they apply the algorithm in two examples to compute the recommendations of movies and intelligent information.

Bonatti *et al.* [4] focus on data trust based on annotations. They identify several annotation dimensions: 1. *Blacklisting*, which is based on noise, on void values for inverse functional properties, and on errors in values; 2. *Authoritativeness*, which is based on cross-defined core terms that can change the inferences over those terms that are mandated by some authority (e.g., owl:Thing), and that can lead to creation of irrelevant data; 3. *Linking*, which is based on determining the existence of links from and to a source in a graph, with a premise that a source with higher number of links is more trustworthy and is characterized by higher quality of the data.

Later on, we can find research work focused on various other aspects of Linked Data quality such as accuracy, consistency, dynamicity, assessibility. Furber and Hepp [9] focus on the assessment of accuracy, which includes both syntactic and semantic accuracy, timeliness, completeness, and uniqueness. One measure of accuracy consists of determining inaccurate values using functional dependence rules, while timeliness is measured with time validity intervals of instances and their expiry dates. Completeness deals with the assessment of the completeness of schema (representation of ontology elements), completeness of properties (represented by mandatory property and literal value rules), and completeness of population (representation of real world entities). Uniqueness refers to the assessment of redundancy, i.e., of duplicated instances.

Flemming [8] focuses on a number of measures for assessing the quality of Linked Data covering wide-range of different dimensions such as availability, accessibility, scalability, licensing, vocabulary reuse, and multilingualism. Hogan *et al.* [15] focus their work in assessment of mainly errors, noise and modeling issues. Lei *et al.* [22] focus on several types of quality problems related to accuracy. In particular, they evaluate incompleteness, existence of duplicate instances, ambiguity, inaccuracy of instance labels and classification.

Rula *et al.* [33] start from the premise of dynamicity of Linked Data and focus on assessment of timeliness in order to reduce errors related to outdated data. To measure timeliness, they define a currency metric which is calculated in terms of differences between the time of the observation of data (current time) and the time when the data was modified for the last time. Furthermore, they also take into account the difference between the time of data observation and the time of data creation.

Gueret *et al.* [14] define a set of network measures for the assessment of Linked Data mappings. These measures are: 1. Degree; 2. Clustering coefficient; 3. Centrality; 4. SameAs chains; 5. Descriptive richness.

Mendes *et al.* [23] developed a framework for Linked Data quality assessment. One of the peculiarities of this framework is to discover conflicts between values in different data sources. To achieve this, they propose a set of measures for Linked Data quality assessment, which include: 1. Intensional completeness; 2. Extensional completeness; 3. Recency and reputation; 4. Time since data modification; 5. Property completeness; 6. Property conciseness; 7. Property consistency.

Kontokostas *et al.* [21] developed a test-driven evaluation of Linked Data quality in which they focus on coverage and errors. The measures they use are the following: 1. Property domain coverage; 2. Property range coverage; 3. Class instance coverage; 4. Missing data; 5. Mistypes; 6. Correctness of the data.

Knuth *et al.* [20] identify the key challenges for Linked Data quality. As one of the key factors for Linked Data quality they outline validation which, in their opinion, has to be an integral part of Linked Data lifecycle. Additional factor for Linked Data quality is version management, which can create problems in provenance and tracking. Finally, as another important factor they outline the usage of popular vocabularies or manual creating of new correct vocabularies.

Emburi *et al.* [6] developed a framework for automatic crawling the Linked Data datasets and improving dataset quality. In their work, the quality is focused on errors in data and the purpose of developed framework is to automatically correct errors.

Assaf *et al.* [1] introduce a framework that handles issues related to incomplete and inconsistent metadata quality. They propose a scalable automatic approach for extracting, validating, correcting and generating

descriptive linked dataset profiles. This approach applies several techniques in order to check the validity of the metadata provided and to generate descriptive and statistical information for a particular dataset or for an entire data portal.

Debattista *et al.* [5] describes a conceptual methodology for assessing Linked Datasets, proposing Luzzu, a framework for Linked Data Quality Assessment. Luzzu is based on four major components: 1. An extensible interface for defining new quality metrics; 2. An interoperable, ontology-driven back-end for representing quality metadata and quality problems that can be re-used within different semantic frameworks; 3. Scalable dataset processors for data dumps, SPARQL endpoints, and big data infrastructures; 4. A customisable ranking algorithm taking into account user-defined weights.

Zaveri *et al.* [38] present a comprehensive systematic review of data quality assessment methodologies applied to LOD. They have extracted 18 quality dimensions and a total of 110 objective and subjective quality indicators.

There is a significant effort in the Semantic Web community to evaluate the quality of Linked Data. However, in the current state of the art, less focus has been given toward understanding knowledge base resource changes over time to detect anomalies over various releases. Overall, the main contribution of our approach is to identify quality issues using change detection over KB releases.

3. Quality Characteristics and Temporal Analysis

Data quality is a cross-disciplinary and multidimensional concept. Data quality, in general, relates to the perception of the "fitness for use" in a given context [35]. According to Pipino *et al.* [29], based on context, quality can be both subjective perceptions and objective measurements. This means that data quality is dependent on the actual use case.

In our approach, we use two data quality standard reference frameworks: ISO/IEC 25012 [17] and W3C DQV [16]. ISO/IEC 25012 [17] defines a general data quality model for data retained in a structured format within a computer system. This model defines the quality of a data product as the degree to which data satisfies the requirements set by the product owner organization. The W3C Data on the Web Best Practices Working Group has been chartered to create a vocabulary for expressing data quality[1]. The Data Quality Vocabulary (DQV) is an extension of the DCAT vocabulary[3]. It covers the quality of the data, how frequently is it updated, whether it accepts user corrections, and persistence commitments.

Table 1
Measurement terminology

Definition	ISO 25012	W3C DQV
Category of quality attributes	Characteristic	Dimension
Variable to which a value is assigned as the result of a measurement function applied to two or more measure elements	Measure	Metric
Variable defined in terms of an attribute and the measurement method for quantifying it	Measure Element	-
Numerical value that characterize a quality feature	Value	Observation
Set of operations having the object of determining a value of a measure	Measurement	Measurement

Besides, to further compare our selected quality characteristics[4] we explored the foundational work on the linked data quality by Zaveri *et al.* [38]. They surveyed existing literature and identified a total of 18 different data quality dimensions (criteria) applicable to linked data quality assessment.

Since the measurement terminology suggested in the two standards is different, we briefly summarize the one adopted in this paper and the relative mapping as reported in Table 1.

3.1. Quality Issues

A data quality issues is a set of anomalies that can affect the potentiality of the applications that use the data [3]. We can identify quality issues through data quality measure. A data quality measure variable to which a value is assigned as the result of measurement of data quality characteristic [17]. Assessing the quality of data usually requires a large number of quality measures to be computed [5]. Each of the quality characteristics identifies a specific set of quality issues. In this paper, we focused on three main quality issues of

[3] https://www.w3.org/TR/prov-o

[4] In our work we will identify the quality aspects using the term quality characteristics from ISO-25012 [17] that corresponds to the term quality dimension from DQV [16].

a knowledge baase such as *(i)* Consistency, *(ii)* Completeness, and *(iii)* Persistency.

Consistency relates to a fact being inconsistent in a KB. In particular, inconsistency relates to the presence of unexpected properties.

As an example let us consider a DBpedia resource of type *foaf:Person*: *X. Henry Goodnough*[5]. We find as expected a *dbo:birthDate* property, but we unexpectedly find the property *dbo:Infrastructure/length*. This is a clear inconsistency: in fact according to the ontology we can expect the latter property for a resource of type *dbo:Infrastructure*, not for a person.

Fig. 1. Example of inconsistent Wikipedia data.

To better understand where the problem lies, we need to look at the corresponding Wikipedia page[6]. Even though the page reports the information about an engineer who graduated from Harvard, it contains an info-box, shown in Figure 1, that refers to a dam, the Goodnough Dike. The inconsistency issue derives from the data present in the source page that resulted into the resource being typed both as a person and as a piece of infrastructure. We can expect such kind of structure to be fairly rare – in fact the case we described is the only case of a person with a *dbo:Infrastructure/length* property – and can be potentially detected by looking at the frequency of the predicates within a type of resource. For instance for the resources of type *foaf:Person* there are 1035 distinct predicates, among which 142 occur only once for DBpedia version 201604.

Completeness relates to the resources or properties missing from a knowledge base. This happens when information is missing or has been removed.

As an example, let us consider a DBpedia resource of type *dbo:Person/Astronauts*: *Abdul Ahad Mohmand*[7]. When looking at the source Wikipedia page[8], we observe that, as shown in Figure 2, the infobox reports a "Time in space" datum. The DBpedia ontology includes a *dbo:Astronaut/TimeInSpace* and several other astronauts have that property, but the resource we consider is missing it.

Fig. 2. Example of incomplete Wikipedia data.

While it is generally difficult to spot that kind of incompleteness, for the case under consideration it is easier because that property was present for the resource under consideration in the previous version of DBpedia, i.e. the 2015-10 release. It is an incompleteness introduced by the evolution of the knowledge base. It can be spotted by looking at the frequency of predicates inside a resource type. In particular, in the release of 2016-04 there are 419 occurrences of the *dbo:Astronaut/TimeInSpace* predicate over 634 astronaut resources, while in the previous version they were 465 out of 650 astronauts.

[5] http://dbpedia.org/resource/X._Henry_Goodnough

[6] https://en.wikipedia.org/wiki/X._Henry_Goodnough

[7] http://dbpedia.org/resource/Abdul_Ahad_Mohmand

[8] https://en.wikipedia.org/wiki/Abdul_Ahad_Mohmand

Persistency relates to resources that were present in a previous KB release but they disappeared. This happens when information has been removed. As an example let us consider a 3cixty Nice resource of type *lode:Event* that has as label the following: "Modéliser, piloter et valoriser les actifs des collectivités et d'un terrritoire grâce aux maquettes numériques: retours d'expériences et bonnes pratiques"[9]. This resource happened to be part of the 3cixty Nice KB since it has been created the first time, but in a subsequent release it got removed even though it should not have been removed.

Subject Item
n2:006dc982-15ed-47c3-bf6a-a141095a5850
rdf:type
lode:Event
rdfs:label
Modéliser, piloter et valoriser les actifs des collectivités et d'un terrritoire grâce aux maquettes numériques : retours d'expériences et bonnes pratiques
rdfs:seeAlso
n13:en
cixty:descriptionScore
0.0
cixty:posterScore
1.0
lode:poster
n4:006dc982-15ed-47c3-bf6a-a141095a5850
dc:identifier
MN13
dc:publisher
n14:com
locationOnt:businessType
n15:event
lode:atPlace
n12:be7fac75-bb59-41fd-a626-4bd7e77f0a7f
lode:atTime
n6:interval
lode:hasCategory
Conferences Maquette Numérique
lode:inSpace
n6:geometry
lode:involvedAgent
n11:a40c9900f85a517cef40ef8f1e4289b9 n11:7f1a9cc96861920e147505e23ea4f913
n11:dce31cbcfdad5c0a180fb4d0efd0c511
locationOnt:cell
n9:1301

Fig. 3. Example of a 3cixty Nice KB resource that unexpectedly disappeared from the release of 2016-06-15 to the other 2016-09-09.

Such a problem is generally complex to be traced manually because it requires a per-resource check over the different releases. It can, instead, be spotted by looking at the total frequency of entities of a given resource type. In particular in the investigated example taken from the 3cixty Nice KB released on 2016-09-09, we have observed an unexpected drop of resources of the type event from the previous release dated as 2016-06-15, which has triggered further investigations.

[9] http://data.linkedevents.org/event/006dc982-15ed-47c3-bf6a-a141095a5850

3.2. Temporal Analysis

Knowledge bases are often maintained by large communities that act as curators to ensure their quality [37]. Knowledge base changes can be categorized as follows: *i)* resource representations and links that are created, updated and removed; *ii)* the entire graph can change or disappear [31]. The kind of evolution that a KB is subjected to depends on several factors, such as:

- Frequency of update: KBs can be updated almost continuously (e.g. daily or weekly) or at long intervals (e.g. yearly);
- Application area: depending on the specific domain, updates can be minor or substantial. For instance, social data is likely to change more frequently than encyclopedic data;
- Data acquisition: the process used to acquire the data to be stored in the KB and the characteristics of the sources may influence the evolution; For instance, updates on individual resources cause minor changes when compared to a complete reorganization of a data source's infrastructure such as a change of the domain name;
- Link between data sources: when multiple sources are used for generating a KB, the alignment and compatibility of such sources affect the overall KB evolution. The differences of KBs have been proved to play a crucial role in various curation tasks such as the synchronization of autonomously developed KB versions, or the visualization of the evolution history of a KB [27] for more user-friendly change management.

3.3. Temporal-based Quality characteristics and Measures

In this section, we define four temporal quality characteristics that allow addressing the aforementioned issues.

Zaveri et al. [38] classified quality dimensions into four groups: *i)* intrinsic, those that are independent of the users context; *ii)* contextual, those that highly depend on the context of the task at hand, *iii)* representational, those that capture aspects related to the design of the data, and *iv)* accessibility, those that involve aspects related to the access, authenticity and retrieval of data obtain either the entire or some portion of the data (or from another source) for a particular use case. The quality dimensions we propose fall into the groups

of intrinsic and representational. Our approach focuses on two different types of elements in a KB: subjects and predicates. The objects, either resources or literals, are not considered. Concerning the subjects we consider them collectively by grouping according to the class – or a property defined as *rdf:type* – they belong to. As far as properties are concerned, we analyze separately the properties of the resources of a given class.

Table 2 reports the proposed characteristics, along with the quality issue they address, the level of measure – either class subject or property –, and the corresponding quality characteristic as defined in the ISO 25012 standard.

Table 2
Quality Characteristics in KB evolution.

Quality Issues	ISO/IEC 25012	Levels	Quality Characteristics
Persistency	Credibility	Class	Persistency
Persistency	Efficiency	Class	Historical Persistency
Completeness	Completeness	Property	Completeness
Consistency	Consistency	Class & Property	Consistency

In order to measure the degree to what extent a certain data quality characteristics is fulfilled for a given KB, each characteristics is formalized and expressed in terms of a measure with a value in the range [0, 1]. We call this measurement function for a data quality characteristics.

3.3.1. Basic Measure Elements

In our approach, we consider changes at the statistical level in terms of variation of absolute and relative frequency count of subjects and predicates between pairs of KB versions.

Our approach shares the same basis as Papavasilieiu *et al.* [27]. They divided the changes into *Low-Level* and *High-Level*. In our evaluation approach, we focus on *Low-Level* changes that consist in the addition or deletion of a triple from a KB.

In particular we aim to detect changes in two basic statistical measures that can be computed with the most simple operation, i.e. counting. The computation is performed on the basis of the classes in a KB (V), i.e. given a class C we consider all the triples t whose subjects have the type C.

The first measure element we define is the count of the instances of a class C:

$$\text{count}(C) = |\{s : \exists \langle s, \text{typeof}, C\rangle \in V\}|$$

The count(C) measurement can be performed by means of a basic SPARQL query such as:

```
SELECT COUNT(DISTINCT ?s) AS ?COUNT
WHERE {  ?s a <C> . }
```

The second measure element focuses on the frequency of the predicates, within a class C. We define the frequency of a predicate (in the scope of class C) as:

$$\text{freq}(p, C) = |\{\langle s, p, o\rangle \in V : \exists \langle s, \text{typeof}, C\rangle \in V\}|$$

The freq(p, C) measurement can be performed by means of a simple SPARQL query having the following structure:

```
SELECT COUNT(*) AS ?FREQ
WHERE {
    ?s <p> ?o.
    ?s a <C>.
}
```

There is an additional basic measure element that can be used to build derived measures: the number of predicates present for the subject class C in the release i of the KB.

$$NP(C) = |\{p : \exists \langle s, p, o\rangle \in V \wedge \langle s, \text{typeof}, C\rangle \in V\}|$$

The $NP(C)$ measure can be collected by means of a SPARQL query having the following structure:

```
SELECT COUNT(DISTINCT ?p) AS ?NP
WHERE {
    ?s ?p ?o.
    ?s a <C>.
}
```

The essence of the proposed approach is the comparison of distinct releases of a KB with respect to subject count or predicate frequency measures. We will use a subscript to indicate the release the measure refers to. The releases are numbered progressively as integers and, by convention, the most recent release is

n. So, for instance, $count_{n-1}(foaf{:}Person)$ represents the count of subjects typed with *foaf:Person* in the last release of the knowledge base under consideration.

3.3.2. Persistency

This quality characteristic relates to the Credibility quality characteristic in the ISO/IEC 25012 standard. Credibility is the "degree to which data has attributes that are regarded as true and believable by users in a specific context of use. Credibility includes the concept of authenticity (the truthfulness of origins, attributions, commitments)" [17]. Considering the correspondence presented in W3C DQV, Zaveri et al. [38] implies Credibility as Trustworthiness. They report that "Trustworthiness is defined as the degree to which the information is accepted to be correct, true, real and credible."

An additional important feature to be considered when analyzing knowledge base is that the information stored is expected to grow, either because of new facts appearing in the reality, as time passes by, or due to an extended scope coverage [36]. Persistency measures provides an indication of the adherence of a knowledge base to such continuous growth assumption. Using this quality measure, data curators can identify the classes for which the assumption is not verified.

The *Persistency* of a class C in a release $i : i > 1$ is defined as:

$$Persistency_i(C) = \begin{cases} 1 \text{ if } count_i(C) \geq count_{i-1}(C) \\ 0 \text{ if } count_i(C) < count_{i-1}(C) \end{cases}$$

the value is 1 if the count of subjects of type C is not decreasing, otherwise it is 0.

Persistency at the knowledge base level, i.e. when all classes are considered, can be computed as the proportion of persistent classes:

$$Persistency_i = \frac{\sum_{j=1}^{NC} Persistency_i(C_j)}{NC}$$

where NC is the number of classes analyzed in the KB.

3.3.3. Historical Persistency

This quality characteristic relates to the Efficiency quality characteristic defined in the ISO/IEC 25012 standard. Efficiency is defined as the "degree to which data has attributes that can be processed and provide the expected levels of performance by using the appropriate amounts and types of resources in a specific context of use" [17].

Considering the mapping presented in W3C DQV, Zaveri et al. [38] implies Efficiency as Performance. They define as, "Performance refers to the efficiency of a system that binds to a large dataset, that is, the more performing a data source is the more efficiently a system can process data".

Historical persistency is a derived measurement function using the persistency measure over all releases of KB. Historical persistency dimensions explore entire KB evolution for a specific entity to detect inconsistency. This metric extends the persistency metric to provide insights on the series of KB releases. It considers all entities presented in a KB and give an overview of the KB. Data curators can get an overview of knowledge base persistency issues over all releases. It helps data curators to decide which knowledge base release can be used for future data management tasks.

The Historical Persistency measure evaluates the persistency over the history of the KB and is computed as the average of the pairwise persistency measures for all releases.

$$\text{H\_Persistency}(C) = \frac{\sum_{i=2}^{n} \text{Persistency}_i(C)}{n-1}$$

Similarly to Persistency, it is possible to compute Historical Persistency at the KB level:

$$\text{H\_Persistency} = \frac{\sum_{i=2}^{n} \text{Persistency}_i}{n-1}$$

3.3.4. Consistency

This quality characteristic relates to ISO/IEC 25012 standard Consistency quality characteristic.

The Consistency is defined as the "degree to which data has attributes that are free from contradiction and are coherent with other data in a specific context of use. It can be either or both among data regarding one entity and across similar data for comparable entities" [17].

Considering the correspondence presented in W3C DQV, *Zaveri et al.* mention Consistency as Conciseness. They define as, "Conciseness refers to the minimization of redundancy of entities at the schema and the data level."

We assume that extremely rare predicates are potentially inconsistent, see e.g. the *dbo:Infrastructure/length* property discussed in the example presented in Section 3.1. We can evaluate the consistency of a predicate on the basis of the frequency basic measure.

We define the consistency of a property p in the scope of a class C:

$$Consistency_i(p,C) = \begin{cases} 1 & \textit{if } \text{freq}_i(p,C) \geq T \\ 0 & \textit{if } \text{freq}_i(p,C) < T \end{cases}$$

Where T is a threshold that can be either a KB-dependent constant[10] or can defined on the basis of the count of the scope class, e.g. $T = \text{count}(C)/10000$.

3.3.5. Completeness

The ISO/IEC 25012 defines the Completeness quality characteristic as the "degree to which subject data associated with an entity has values for all expected attributes and related entity instances in a specific context of use" [17].

In Zaveri *et al.*, Completeness refers to the degree to which all required information is present in a particular dataset. In terms of Linked Data, completeness comprises of the following aspects: *i)* Schema completeness, the degree to which the classes and properties of an ontology are represented, thus can be called "ontology completeness"; *ii)* Property completeness, measure of the missing values for a specific property, *iii)* Population completeness is the percentage of all real-world objects of a particular type that are represented in the datasets, and *iv)* Interlinking completeness, which has to be considered especially in Linked Data, refers to the degree to which instances in the dataset are interlinked.

Temporal-based completeness focuses on the removal of information as a negative effect of the KB evolution. It is based on the continuous growth assumption as well; as a consequence we expect properties of subjects should not be removed as the KB evolves (e.g. *dbo:Astronaut/TimeInSpace* property described in the example presented in Section 3.1).

The basic measure we use is the frequency of predicates, in particular, since the variation in the number of subjects can affect the frequency, we introduce a normalized frequency as:

$$\text{Nf}_i(p,C) = \frac{\text{freq}_i(p,C)}{\text{count}_i(C)}$$

On the basis of this derived measure we can thus define completeness of a predicate p in the scope of a class C as:

$$Completeness_i(p,C) = \begin{cases} 1, & \text{Nf}_i(p,C) \geq \text{Nf}_{i-1}(p,C) \\ 0, & \text{Nf}_i(p,C) < \text{Nf}_{i-1}(p,C) \end{cases}$$

At the class level the completeness is the proportion of complete predicates and can be computed as:

$$Completeness_i(C) = \frac{\sum_{k=1}^{\text{NP}_i(C)} Completeness_i(p_k,C)}{\text{NP}_i(C)}$$

where $NP_i(C)$ is the number of predicates present for the subject class C in the release i of the knowledge base, and p_k.

4. Temporal-based Quality Assessment Approach

Data quality life cycle generally includes the identification of quality requirements and relevant metrics, quality assessment, and quality improvement [5] and [26]. Debattista *et al.* [5] present a data quality life cycle that covers the phases from the assessment of data, to cleaning and storing. They show that in the lifecycle quality assessment and improvement of Linked Data is a continues process. However, we explored the features of quality assessment based on KB evolution.

We argue that a KB development and maintenance follows a similar Data Life Cycle as standard software development. In standard software development, the Data Life Cycle (DLC) provides a high level overview of the stages involved in a successful management and preservation of data for any use and reuse process. In particular, multiple versions of a data life cycle exist with differences to the attributable to variation in practices across domains or communities [3]. Our reference Data Life Cycle is defined by the international standard ISO 25024 [17], which we extend and represented in Figure 4 where we integrate a quality assessment phase along with the data collection, data integra-

[10] In our experiments we used T=100 as empirically verified to maximize the precision of the approach in detecting quality issues.

tion, and external data acquisition phase. This phase ensures data quality for the data processing stage. The first step in building the quality assessment approach was to identify the quality characteristics.

Based on the quality characteristics presented in Section 3.3, we proposed a KB quality assessment approach describing, in detail, our temporal quality assessment approach that computes statistical distributions of KB elements from different KB releases and, when observing differences in patterns, it detects anomalies. Figure 5 frames our quality assessment approach as a three-stage process: *(1)* input data, *(2)* evaluation process, and *(3)* quality reporting. The various stages are explained in detail below.

1. **Input data** Different releases of the same knowledge bases chronologically sorted constituting a vector. This vector can be acquired via querying multiple SPARQL endpoints (assuming each endpoint hosts one release of the knowledge base, or querying only on specifying the time of the release as constraint) or via loading data dumps. We also assume that each KB release is composed of triples of the form (s, p, o), where s is the subject, p the predicate, and o the object of a statement. p includes also the resource class belonging to a pre-defined ontology. In our approach an intermediary data structure has been created to group together resources belonging to a particular class. We build an intermediary data structure by grouping together set of resources and predicates for each class based on KB releases, where the class is $(c_1...c_n)$, with n equal to the cardinality of all resource classes in p. We perform statistical operations for each release of the knowledge base grouping together subjects by classes; this helps the implementation of measurement functions as we can perform all computations on the intermediary data structure. In Figure 6, we illustrate the intermediary data structure that is used as input of the next stage.
2. **Evaluation Process** Anomalies can be identified by a data curator using data profiling and statistical analysis techniques. Relying on data-driven measurements of changes over time, the evaluation process of the knowledge base quality is thus performed based on quality characteristics presented in Section 3.3. Each quality characteristic indicates a specific quality measure. In our approach, quality characteristics are measured by using statistical information from each knowledge base release under assessment. We define this step as quality profiling and it includes statistical as well as quality characteristics measurement functions. In particular, quality characteristic measurement is done by examining a knowledge base release. Finally, quality assessment will result in a record of quality information for each assessed knowledge base release. This generates a quality problem report, which can be as detailed as pinpointing specific problems at the level of individual triples.
The evaluation process includes the following two steps:

   (a) **Statistical Profiler:** For computing the change detection in a vector of KB releases, we used basic statistical operations. We thereby use the following key statistics:
   (i) number of distinct predicates; *(ii)* number of distinct subjects; *(iii)* number of distinct entities per class; *(iv)* frequency of predicates per entity ;
   To identify the KB release changes, we count the frequency of property values for a specific class. Also, we consider the distinct entity count for a specific class that we presented as measurement elements in Section 3.3.1. From the summary statistics, in our approach, we used frequency of the properties and distinct entity count as key statistical elements for change detection operations in a quality characteristics measure. In particular, we compute change detection between two KB releases by observing the variation of key statistics. In Table 2, we divide our quality characteristics in class and property level. For class level quality characteristics, we considered entity count as the basic measurement elements for change detection. For a particular class, we measure the property level quality characteristics using frequency of properties as basic measurement elements for change detection.
   In the evaluation process, the statistical profiler outputs are used by the quality profiler as inputs.
   (b) **Quality Profiler:** Data profiling is defined as the process of creating descriptive information and collect statistics about the data [1]. It summarizes the dataset without inspecting the raw data. We used the ap-

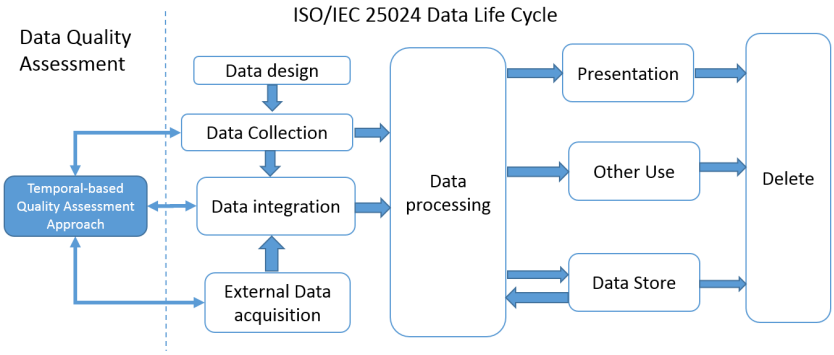


Fig. 4. ISO/IEC 25024 Data Life Cycle [17] with proposed quality assessment approach.

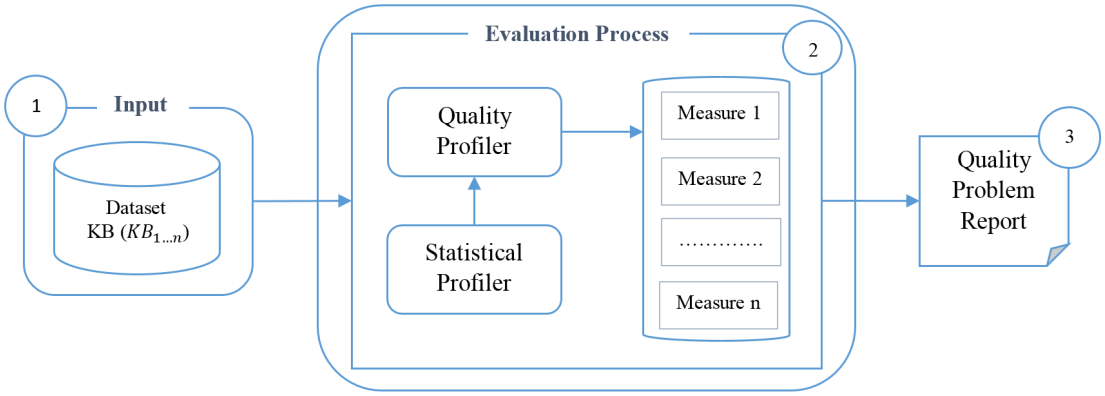


Fig. 5. Workflow of the proposed Quality Assessment Approach.

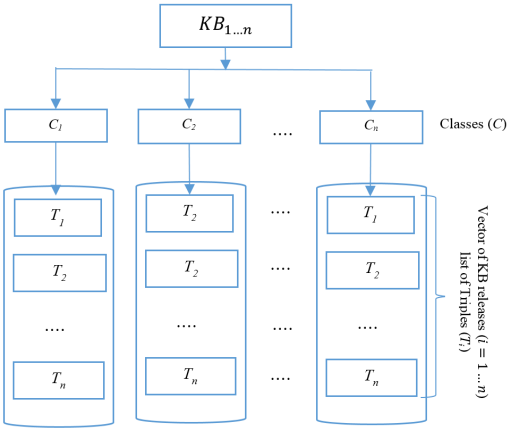


Fig. 6. Intermediary data structure that is used as input for the Evaluation Process.

proach of data profiling together with quality measure to profile quality issues. We used statistical profiler to analyze the KB releases. For analyzing the KB datasets, we used four quality characteristics presented in Section 3.3. Quality profiler includes descriptive as well as measure values based on the quality characteristics.

More in detail, this component does the following tasks: *(i)* it provides statistical information about KB releases and patterns in the dataset (e.g. properties distribution, number of entities and RDF triples); *(ii)* it provides general information about the KB release vector, such as dataset description of class and properties, release or update dates; *(iii)* it provides quality information about the vector of KB releases, such as quality measure values, list of erroneous analysed triples.

3. **Quality Reporting** We generate a quality report based on the quality assessment results. The reports contain quality measures computation results as well as summary statistics of each class.

Quality problem report provides detailed information about erroneous classes and properties. Also, the quality measurement results can be used for cataloging and preservation of the knowledge base for future data curation tasks. In particular, the Quality Problem Reporting enables, then, a fine-grained description of quality problems found while assessing a knowledge base.

5. Experimental Analysis

This section reports an experimental analysis of our approach on two KBs, namely DBpedia and 3cixty Nice. The analysis is based on a prototype implementation. We first present the distinct building blocks of the implementation and then we report the results of both *(i)* a quantitative and *(ii)* a qualitative validations.

5.1. Experimental Settings

In our experiments, we selected two KBs according to: *i)* popularity and representativeness in their domain: DBpedia for the encyclopedic domain, 3cixty Nice for the tourist and cultural domain; *ii)* heterogeneity in terms of content being hosted, *iii)* diversity in the update strategy: incremental and usually as batch for DBpedia, continuous update for 3cixty Nice. More in detail:

- *DBpedia*[11] is among the most popular knowledge bases in the LOD cloud. This knowledge base is the output of the DBpedia project that was initiated by researchers from the Free University of Berlin and the University of Leipzig, in collaboration with OpenLink Software. DBpedia is roughly updated every year since the first public release in 2007. DBpedia is created from automatically-extracted structured information contained in Wikipedia[12], such as infobox tables, categorization information, geo-coordinates, and external links.
- *3cixty Nice* is a knowledge base describing cultural and tourist information concerning the city of Nice. This knowledge base was initially developed within the 3cixty project[13], which aimed to develop a semantic web platform to build real-world and comprehensive knowledge bases in the domain of culture and tourism for cities. The entire approach has been tested first in the occasion of the Expo Milano 2015 [7], where a specific knowledge base for the city of Milan was developed, and has now been refined with the development of knowledge bases for the cities of Nice, London, Singapore, and Madeira island. They contain descriptions of events, places (sights and businesses), transportation facilities and social activities, collected from numerous static, near- and real-time local and global data providers, including Expo Milano 2015 official services in the case of Milan, and numerous social media platforms. The generation of each city-driven 3cixty KB follows a strict data integration pipeline, that ranges from the definition of the data model, the selection of the primary sources used to populate the knowledge base, till the data reconciliation used for generating the final stream of cleaned data that is then presented to the users via multi-platform user interfaces. The quality of the data is today enforced through a continuous integration system that only verifies the integrity of the data semantics [26].

5.2. Implementation

Our approach has been implemented with a prototype written in R[14]; the design consists of four main components; more in details:

1) **Input:** The input block consists of components to load KB releases and sort them by time. We extracted each version of the knowledge base separately via querying a dedicated SPARQL endpoint and put together into a vector of releases. We saved the results extracted from the SPARQL endpoints into CSV file. We named each CSV file based on the knowledge base release and class name. To build an intermediate data structure, we performed grouping of CSV files using the filename. In Figure 7, we illustrate the class base grouping of extracted CSV files for all DBpedia KB releases. For instance, we extracted all triples of the 11 DBpedia KB releases belonging to the subjects having as class *foaf:Person* and saved in a CSV file with DBpedia release as filename. In particular, the input module processes a collection of CSV files where each file is extracted from the KB release.

[11] http://wiki.dbpedia.org
[12] https://www.wikipedia.org
[13] https://www.3cixty.com
[14] https://github.com/rifat963/KBQ

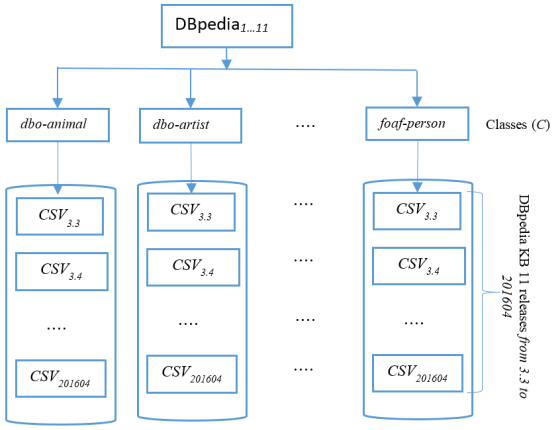


Fig. 7. Structure of input module.

In Figure 6, we present an overview of our intermediate data structure after processing. The processing consists of the following steps: *(i)* compute the distribution of classes in a release; *(ii)* compute the distribution of properties of a given class. We create a vector for each class. This vector is chronologically sorted according to the KB releases dates. *(iii)* for a specific class if the property is not present in all KB releases, we then remove the property from the table. The removal of a property not present in all releases will ensure the schema level consistency.

2) **Statistical Profiler:** We implemented statistical profiler based on the basic measurement elements presented in Section 3.3.1. The intermediate data structure provided by the input block piped through a statistical analysis component. In particular, this component based on two statistical functions, the number of distinct entities per class and frequency of predicates per entity. We used the results from statistical analysis component in the change detection function for the quality characteristics measure. For instance, we extracted all triple of *lode:Event* class of 3cixty Nice KB 8 releases. In the input block, we processed *lode:Event* class to build a vector based on each releases and piped through the statistical profiler to compute summary statistics. We used summary statistics as input to the quality profiler.

3) **Quality Profiler:** This component divided into two modules: first, a module that computes the statistics of the triples by class; second, a module that computes the quality measures. We implemented the first module using the statistical profiler. It provides summary statistics on the number of distinct entity count and frequency of predicates per entity. The second module based on the quality measure present in the section 3.3. In particular, using the summary statistics as basic measurement elements presented in Section 3.3.1 we compute the change detection between KB releases. Using the change detection function we computed our proposed four quality characteristics present in the Section 3.3.

4) **Quality problem report:** The results provided by the Quality Profiler block are then processed by this component to visualize the quality measures and summary statistics. We implemented quality problem report visualization in a R markdown documents. R markdown documents are fully reproducible and easy to demonstrated results with graphs and tables.

5.3. Quantitative Analysis

We applied our quantitative analysis approach based on the quality characteristic measurement to the 3cixty Nice and DBpedia KBs. We analyzed a few classes from the two KBs to investigate persistency, historical persistency, consistency, and completeness quality characteristics. The goal was to identify any classes and properties affected by quality issues. In Table 3, we present the interpretation criteria for each quality characteristic measure. In the remainder of this section, for each knowledge base we first present the experimental results from the analysis, then we then discuss them.

5.3.1. 3cixty Nice Case Study

For this case study, we considered eight different releases of the 3cixty Nice KB: from 2016-03-11 to 2016-09-09. We considered two subjects having the *rdf:type*[15] *lode:Event*[16] and *dul:Place*[17]. The distinct instance count for each class is presented in Table 4. The variation of *count* in the dataset and the observed history is presented in Figure 8. From the 3cixty Nice KB, we collected a total of 149 distinct properties for the *lode:Event* typed entities and 192 distinct properties for the *dul:Place* typed entities across eight different releases.

Persistency. According to the definition of Persistency (Section 3.3), we computed it over the last two releases of the 3cixty Nice KB. In Table 4, we highlight the last two releases with a separator line. In case

[15] https://www.w3.org/1999/02/22-rdf-syntax-ns#type
[16] http://linkedevents.org/ontology/Event
[17] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Place

Table 3
Quality measures verification conditions

Quality Characteristics	Level	Measure	Interpretation
Persistency	Class	Persistency measure values [0,1]: class specific measure result to identify presistency issue.	The value of 1 implies no persistency issue present in the class. The value of 0 indicates persistency issues found in the class.
	Class	Percentage (%) of persistency: estimation of persistency issue for a specific KB release.	High % presents an estimation of fewer issues, and lower % entail more issues in KB releases.
Historical Persistency	Class	Percentage (%) of historical persistency: estimation of persistency issue over all KB releases	High % presents an estimation of fewer issues, and lower % entail more issues present in KB releases.
Consistency	Property	List of properties with consistency measures value [0,1]: property specific measurement to identify consistency issue.	The value of 1 implies no consistency issue present in the property. The value of 0 indicates consistency issues found in the property.
Completeness	Property	List of properties with completeness measures weighted value [0,1]: property specific measure to detect completeness issue.	The value of 1 implies no completeness issue present in the property. The value of 0 indicates completeness issues found in the property.
	Class	Percentage (%) of completeness: estimation of completeness issue in a class for a specific KB release.	High % presents an estimation of fewer issues, and lower % entail more issues in KB release.

Table 4
3cixty Entity Count

Releases	Time(days)	No. of *lode:Event* entities	No. of *dul:Place* entities
2016-03-11	0	605	20,692
2016-03-22	11	605	20,692
2016-04-09	29	1,301	27,858
2016-05-03	53	1,301	26,066
2016-05-13	63	1,409	26,827
2016-05-27	77	1,883	25,828
2016-06-15	96	2,182	41,018
2016-09-09	182	689	44,968

of *lode:Event* instances, for the KB releases listed in Table 4, we can observe that n^{th} entity $count = 689$ and $(n-1)^{th}$ entity $count = 2182$, where $n = 8$. Based on the persistency measure, $count_n < count_{n-1}$ that implies that the value of $Persistency(lode{:}Event) = 0$. This class presents a persistency issue.

Similarly, in case of *dul:Place* instances, from the dataset we can see that n^{th} entity $count = 44968$ and $(n-1)^{th}$ entity $count = 41018$, where $n = 8$. Based on the persistency measure, $count_n > count_{n-1}$ that implies the value of $Persistency(dul{:}Place) = 1$. Thus, none persistency issue is identified.

We computed 3cixty Nice KB percentage of persistency based on *lode:Events* and *dul:Places* class persistency measure value of $[0, 1]$. The 3cixty Nice KB percentage of Persistency (%)=$(\frac{\text{No. of classes with issues}}{\text{Total no. of classes}}) * 100 = (\frac{1}{2}) * 10 = 50\%$.

Historical Persistency. It focuses on the variations observed across all KB releases. Specifically, the variations of persistency measure are considered only between 2016-06-15 and 2016-09-09 release. We computed it using the persistency measures presented in Table 4. For *lode:Event*-type entities the number of persistency variations with value of 1 present over 8 releases $= 6$.

So the, for the *lode:Event*-type the percentage of historical persistency measure value= $(\frac{6}{7}) * 100 = 85.71\%$.

Similarly, for *dul:Place*, the number of persistency variation with value of 1 present over 8 releases $= 5$. In particular, persistency measure value of 0 presented among four releases, (2016-04-09, 2016-05-3) and (2016-5-13, 2016-05-27). So, for the *dul:Place*-type the historical persistency measure assumes the value= $(\frac{5}{7}) * 100 = 71.42\%$.

Completeness. The measure has been computed based on the last two KB releases, namely 2016-05-15 and 2016-09-09. Based on the definition (Sec 3.3), for

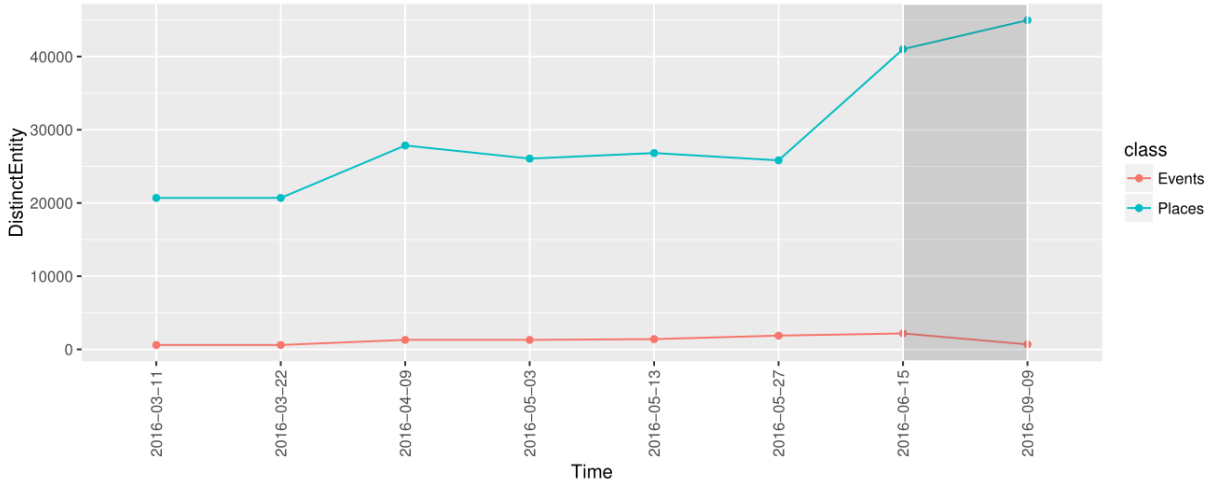


Fig. 8. Variation of instances of 3cixty classes *lode:Event* and *dul:Place* over 8 releases.

the *lode:Event*, the number of predicates in the last two releases $= 21$ and the number of predicates with completeness issues (value of 0) $= 8$. In Figure 9, we report the measure of completeness for the *lode:events*-type where we only present those properties with issues (value of 0).

The percentage of completeness for *lode:Event* class $(\frac{13}{21}) * 100 = 62\%$. Similarly for *dul:Place*, the number of predicates in the last two releases $= 28$ and the number of predicate with completeness issue (value of 0) $= 14$. In Figure 10, we present *dul:Place* class completeness measure results of those properties with completeness issue (value of 0).

The percentage of completeness for the *dul:Place*-type is equal to $(1 - \frac{14}{28}) * 100 = 50\%$

***Consistency*.** This measure identifies the properties with consistency issues for any knowledge base release. Here we focus on the latest release (2016-09-09) of 3cixty Nice KB only. We analyzed *lode:Event*-type and *dul:Place*-type instances. Based on a fixed threshold values of 100[18], we measured the consistency for *lode:Event* and *dul:Place*-type. From the *lode:Event*-type resources, by applying the consistency analysis, we found 10 properties below the fixed threshold. Similarly, for *dul:Place*-type resources we found 12 properties below the threshold value.

Discussion. Table 3 presents an overview of quality measures with the relative interpretation. For both investigated types, the percentage of knowledge base Persistency is 50%, which triggers a warning to a data curator concerning a potential persistency issue existing in the latest (2016-09-09) KB release. For the *lode:Event*-type, *Persistency* $= 0$. More in detail, if we consider the two latest releases (i.e. 2016-06-15, 2016-09-09) of the KB and we filter by the type *lode:Event*, the distinct entity counts are equal to 2182 and 689 respectively. Apparently more than 1400 events disappeared in the 2016-09-09 release: this indicates a potential error in the of 3cixty Nice KB.

The Historical Persistency quality measure provides an overview on the different KB releases. It identifies those versions with issues along the different KB releases. In the case of the 3cixty Nice KB, the *lode:Event* class has one drop (2016-06-15, 2016-09-09) and *dul:Place* class has two (2016-04-09, 2016-05-3), (2016-5-13, 2016-05-27).

The Consistency measure identifies only those properties whose frequency is below a given threshold. In the 3cixty Nice KB latest release (2016-09-09), we only found ten properties for *lode:Event*-type and twelve for *dul:Place*-type resources. We further investigate this output in the qualitative analysis.

The Completeness metric is based on a pairwise comparison of releases. Looking at the two latest releases (2016-06-15, 2016-09-09) of the 3cixty Nice KB, we have identified those properties with completeness value of 0 as issue indicator. The total number of properties of the latest two versions are 21 excluding those properties not presented in both releases. For

[18]This value is empirically assessed by maximizing the precision in error detection as reported in Section 5.4

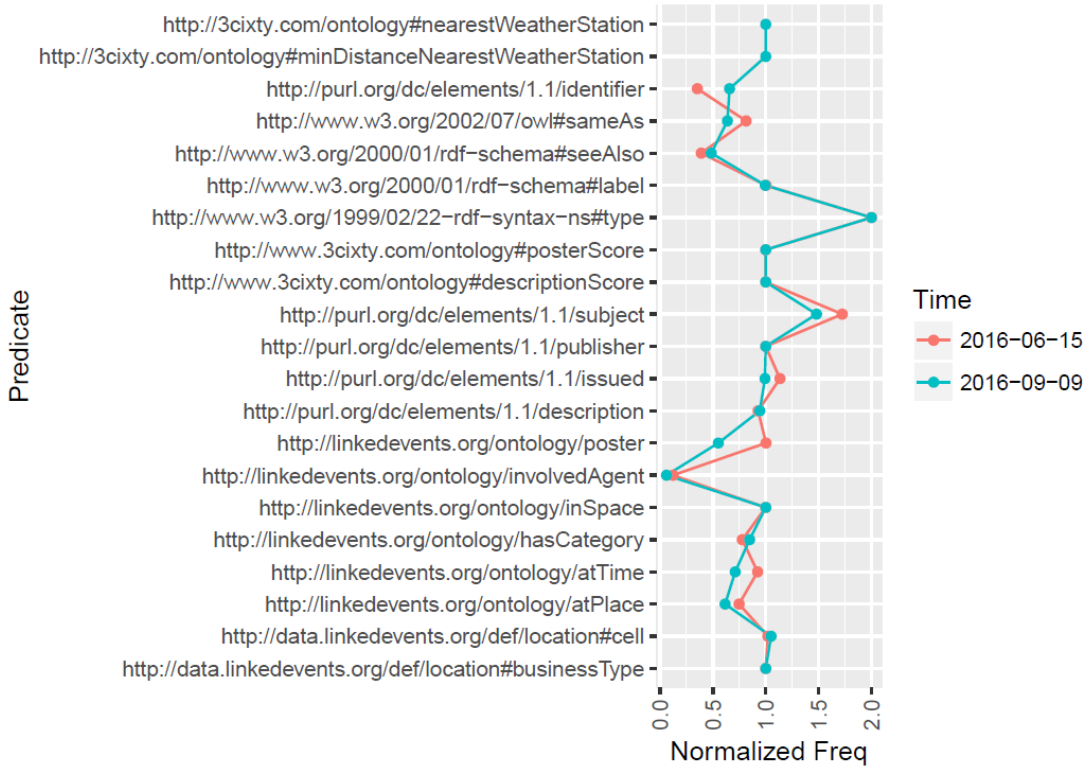


Fig. 9. 3cixty *lode:Event* completeness measure results

instance, the *lode:Event* class property *lode:atPlace*[19] exhibits an observed frequency of 1632 in release 2016-06-15, while it is 424 in release 2016-09-09. As a consequence the Completeness measure evaluates to 0, thus it indicates an issue of completeness in the KB. In 3cixty, the *dul:Place* percentage of completeness is 50%, such a figure indicates a high number of incomplete predicates in the latest version (2016-09-09).

5.3.2. *DBpedia Case Study*

In the case of DBpedia, we considered ten classes: *dbo:Animal*[20], *dbo:Artist*[21], *dbo:Athlete*[22], *dbo:Film*[23], *dbo:MusicalWork*[24], *dbo:Organisation*[25], *dbo:Place*[26], *dbo:Species*[27], *dbo:Work*[28], *foaf:Person*[29]. The above entities are the most common according to the total number of entities. A total of 11 DBpedia releases have been considered for this analysis. We extracted 4477 unique properties from DBpedia. Table 5 presents the breakdown of frequency per class.

For the extraction process we used Loupe [24], an online tool that can be used to inspect and to extract automatically statistics about the entities, vocabularies used (classes, and properties), and frequent triple patterns of a KB. However, we only used Loupe to access the various DBpedia KB releases SPARQL endpoint to extract all triples for the selected 10 classes. Using the extracted data, we built an intermediate data structure as shown in Figure 7.

Persistency. We compare the last two releases (201510, 201604) in terms of entity counts for the ten classes; the two releases are highlighted in Table 5. The resulting Persistency measure values are

[19] http://linkedevents.org/ontology/atPlace
[20] http://dbpedia.org/ontology/Animal
[21] http://dbpedia.org/ontology/Artist
[22] http://dbpedia.org/ontology/Athlete
[23] http://dbpedia.org/ontology/Film
[24] http://dbpedia.org/ontology/MusicalWork
[25] http://dbpedia.org/ontology/Organization
[26] http://dbpedia.org/ontology/Place
[27] http://dbpedia.org/ontology/Species
[28] http://dbpedia.org/ontology/Work
[29] http://xmlns.com/foaf/0.1/Person

Fig. 10. 3cixty *dul:Place* completeness measure results

Table 5
DBpedia 10 Classes entity count

Version	dbo:Animal	dbo:Artist	dbo:Athlete	dbo:Film	dbo:MusicalWork	dbo:Organisation	dbo:Place	dbo:Species	dbo:Work	foaf:Person
3.3	51,809	65,109	95,964	40,310	113,329	113,329	31,8017	11,8042	213,231	29,498
3.4	87,543	71,789	113,389	44,706	120,068	120,068	337,551	130,466	229,152	30,860
3.5	96,534	73,721	73,721	49,182	131,040	131,040	413,423	146,082	320,054	48,692
3.6	116,528	83,847	133,156	53,619	138,921	138,921	413,423	168,575	355,100	296,595
3.7	129,027	57,772	150,978	60,194	138,921	110,515	525,786	182,848	262,662	825,566
3.8	145,909	61,073	185,126	71,715	159,071	159,071	512,728	202,848	333,270	1,266,984
3.9	178,289	93,532	313,730	77,794	198,516	178,516	754,415	202,339	409,594	1,555,597
2014	195,176	96,300	336,091	87,285	193,205	193,205	816,837	239,194	425,044	1,650,315
201504	214,106	175,881	335,978	171,272	163,958	163,958	943,799	285,320	588,205	2,137,101
201510	232,019	184,371	434,609	177,989	213,785	213,785	1,122,785	305,378	683,923	1,840,598
201604	227,963	145,879	371,804	146,449	203,392	203,392	925,383	301,715	571,847	2,703,493

reported in Table 6. The *foaf:Person* entity counts for the two release (201510, 201604) are respectively $(1,840,598 < 2,703,493)$, thus we find no persistency issue. However, Persistency for the remaining nine classes reported in Table 6 is 0 since the entity counts in version 201604 are consistently lower than

version 201510. This implies that when DBpedia was updated from version 201510 to 201604, Persistency issues appeared in the DBpedia for nine classes, the exception being only *foaf:Person*.

Table 6
DBpedia Persistency and Historical Persistency

Class	Persistency latest release	Releases with Persistency = 0	Historical Persistency
dbo:Animal	0	[201604]	89%
dbo:Artist	0	[3.7, 201604]	78%
dbo:Athlete	0	[201504, 3.5, 201604]	67%
dbo:Film	0	[201604]	89%
dbo:MusicalWork	0	[3.7, 2014, 201504, 201604]	56%
dbo:Organisation	0	[2014, 201604]	78%
dbo:Place	0	[201604]	89%
dbo:Species	0	[201604]	89%
dbo:Work	0	[3.7, 201604]	78%
foaf:Person	1	[201510]	89%

Historical Persistency. Figure 11 reports the evolution of the 10 classes over the 11 DBpedia releases investigated in our analysis; the diagram highlights the area corresponding to the latest two versions (201510, 201604). The measurement values are reported in Table 6 in the rightmost column. The results of *dbo:Animal*, *dbo:Film*, *dbo:Place* and *foaf:Person* classes show only one persistency drop over all the releases. However, *dbo:MusicalWork* has four persistency value of 0 over all releases. The *dbo:MusicalWork* class has the highest number of variations over the release which leads to a low historical persistency value of $(\frac{5}{9}) * 100 = 55.55\%$.

Consistency. Table 7 reports, for the DBpedia ten class, the total number of properties, the *consistent* properties – i.e. those with consistency value = 0 –, and the consistent properties – consistency value = 1 –. The values are based on the latest two releases, 201510 and 201604. We measured the consistency identifying those properties with the frequency lower than the threshold value $T = 100$. For example, *foaf:Person* has a total of 436 properties over the 201510 and 201604 releases. We found 198 inconsistent properties, i.e. properties whose frequency is lower than the threshold.

Completeness. Table 8 reports the results of the completeness based on the latest two releases of DBpedia 201510 and 201604. The Completeness measure was applied only to the consistent properties, i.e. properties having a Consistency measure = 1. The table reports, for each class, the total number of properties, the consistent properties – which were considered for completeness computation –, the complete properties, the incomplete properties, and the percentage of complete properties (w.r.t. the consistent ones).

For example, *foaf:Person* has a total of 436 properties over the two considered versions. The number of consistent properties is 238. We computed the completeness measures over those 238 properties and identified 50 properties with completeness measure value of 0 (incomplete). The remaining 188 properties can be considered as complete. The percentage of complete properties can be computed as $(\frac{188}{238}) * 100 = 79.0\%$.

Table 7
Properties for the DBpedia classes and Consistency measures. Results are based on Version 201510 & 201604 with threshold T=100.

Class	Total	Inconsistent	Consistent
dbo:Animal	174	136	36
dbo:Artist	442	348	94
dbo:Athlete	447	304	143
dbo:Film	508	389	389
dbo:MusicalWork	344	297	119
dbo:Organisation	1,143	773	370
dbo:Place	1,194	714	480
dbo:Species	183	99	84
dbo:Work	964	710	254
foaf:Person	436	198	238

Discussion From the analysis conducted using the persistency measure, only the *foaf:Person* class has persistency measure value of 1 indicating no isssue. Conversely, all the remaining nine classes show persistency issues as indicated by a measure value of 0. The DBpedia version with the highest number of inconsistent classes is 201604, with a percentage of persistency is equal to 10%.

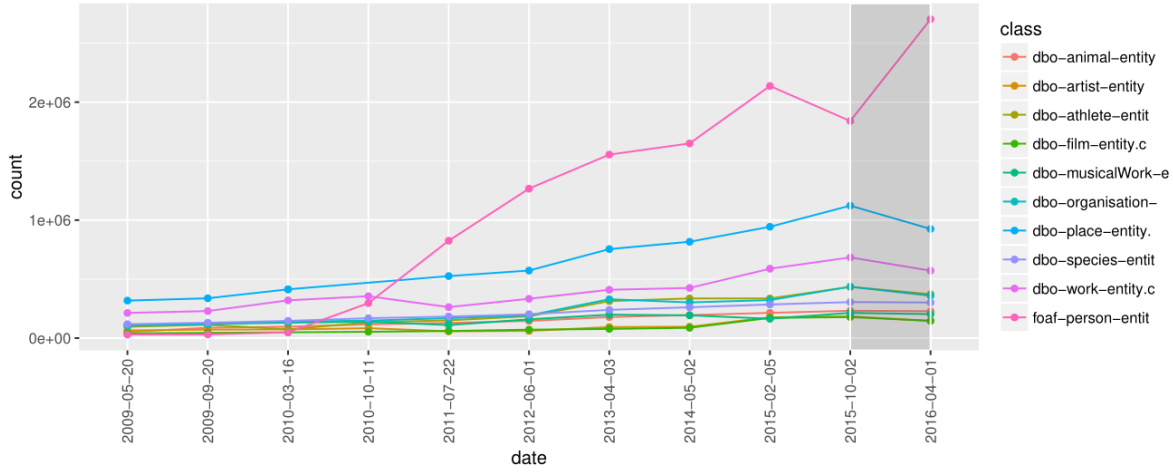


Fig. 11. DBpedia 10 Classes instance variation over 11 releases

Table 8
DBpedia 10 class Completeness measure results based on release 201510 and 201604

Class	Properties	Consistent	Incomplete	Complete	Complete (%)
dbo:Animal	174	38	37	1	2.6%
dbo:Artist	442	94	79	15	15.9%
dbo:Athlete	447	143	124	19	13.3%
dbo:Film	508	389	114	275	70.6%
dbo:MusicalWork	344	119	44	75	63.0%
dbo:Organisation	1,143	370	334	36	9.7%
dbo:Place	1,194	480	437	43	9.0%
dbo:Species	183	84	42	42	50%
dbo:Work	964	254	241	13	5.1%
foaf:Person	436	238	50	188	79.0%

Historical Persistency is mainly an observational measure. Using this measure value, the data curators can study the behaviour of the KB over the different releases. An ideal example is represented by the *foaf:Person* class. From the results, we observe that for the last two releases (201510, 201604) *foaf:Person* is the only class without persistency issues. However, looking at the Historical Persistency results, *foaf:Person* has persistency value of 0 over the releases of 201504 and 201510. Such values may represent a warning to any data curator interested in the past evolution of the KB. From the release 3.3 to 201604, *dbo:MusicalWork* shows the lowest values of persistency as a result the historical persistence is 55.55%.

The Consistency measure identifies only those properties whose frequency is below the threshold value, which triggers a warning to a data curator concerning a potential consistency issue exist. In the latest two releases of the DBpedia KB (201510,201604) we identified consistent properties for 10 classes. Consistency measure results present in Table 7. For example *foaf:Person* class has 198 inconsistent properties. We further investigate this measure through manual evaluation for class *foaf:Person*.

The Completeness measure is based on comparison between any two KB releases. For DBpedia KB looking at the last two releases (201510,201604) we identified incomplete properties for 10 classes. Completeness measure results are listed in Table 8. For instance, we identified a total of 50 incomplete properties for *foaf:Person* class. The *foaf:Person* class property *firstRace*[30] exhibits an observed frequency of 796 in release 201510, while it is 788 in release 201604. As a consequence the Completeness measure evaluated to 0, thus it indicates an issue of completeness in the KB. We further validate our results through manual inspection. In DBpedia, the (dbo:Animal) percentage of completeness is 2.6%, such figure indicates a high number of incomplete predicates in the last release (201604).

5.4. Qualitative Analysis

The general goal of our study is to verify how the *temporal analysis of the changes observed in a set of KB releases helps in quality issue detection*. In the quantitative analysis, we identified classes and properties with quality issues. We, then, summarize on the qualitative analysis based on the results of the quantitative analysis. In particular we carry on a manual investigation of the results obtained by computing the quality measures.

Given the large number of resources and properties, we considered just a few classes and a portion of the entities and properties belonging to those classes in order to keep the amount of manual work to a feasible level. The selection has been performed in a total random fashion to preserve the statistical relevance of the sample. We focused on the effectiveness of the quality measures. A measure is considered effective when it is able to detect an actual problem in the KB. A quality issue identifies a potential error in the KB. In particular, using the interpretation criteria reported in Table 3, from the measure value we can identify a quality issue. The results are a set of potential problems, part of them are accurate – they point to actual problems –, while others are not – they point to false problems –.

We decided to measure precision for evaluating the effectiveness of our approach. Precision is defined as the proportion of accurate results of a quality measures over the total results. More in detail, for a given quality measure, we define an item – either a class or a property – as true positive (*TP*) if, according to the interpretation criteria, the item presents an issue and an actual problem was detected in the KB. An item represents a false positive (*FP*) if the interpretation identifies a possible issue but none actual problem is found. The precision can be computed as follows:

$$p = \frac{TP}{TP + FP}. \quad (1)$$

We evaluated the precision manually by inspecting the results marked as issues from the completeness and consistency measures.

We considered the results obtained by the quantitative analysis for the entities types and properties attached to the class *lode:Event* for the 3cixty Nice KB; we considered entities and properties related to the classes *dbo:Species* and *foaf:Person* for the DBpedia KB. We designed a set of experiments to measure the precision as well as to verify quality characteristics. In Table 9, we present an overview of our selected classes and properties along with the experiments and, in Table 10, we summarize the manual evaluation results.

Table 9
Selected classes and properties for manual evaluation.

KB	Level	Experiment
3cixty Nice	Class	Event class to verify Persistency and Historical Persistency
	Property	*lode:Event* 8 properties from completeness measure to verify and compute precision for completeness.
	Property	*lode:Event* 10 properties from consistency measure to verify and compute precision for consistency.
DBpedia	Class	*dbo:Species*, *foaf:Person* class to verify persistency and historical persistency
	Property	*foaf:Person* class 50 properties from completeness measure to verify and compute precision for completeness.
	Property	*foaf:Person* class 158 properties from consistency measure to verify and compute precision for consistency.

Persistency & Historical Persistency We evaluated persistency measures based on the number of entity counts for *lode:Event* between two different

[30] http://dbpedia.org/ontology/firstRace

Table 10
Summary of manual validation results

Characteristic	3cixty Nice	DBpedia
Persistency	True positive: *lode:Event* entities missing due to algorithm error	False positive; *dbo:Species* class quality issues fixed in current version.
Historical Persistency	True positive: *lode:Event* entities missing due to algorithm error	False positive ; *dbo:Species* class quality issues fixed in current version.
Consistency	False Positive: *lode:Event* properties 2016-09-09 release we did not find any error	True positive ; *foaf:Person* class on 201604 version we identify properties with consistency issue. Based on the threshold value of 100, it has a precision of 68%.
Completeness	True positive: *lode:Event* properties missing due to algorithm error. Over 8 properties we computed Precision of 95% ; True positive	*foaf:Person* properties missing due to completeness issue. Over 50 properties we computed Precision of 94%.

KB releases (2016-06-15, 2016-09-09) of the 3cixty Nice KB. From the quantitative analysis, we detected *lode:Event* has persistency issue with measure value of 0.

For what concerns DBpedia, out of the ten classes under investigation, nine of them have persistency value of 0, which implies that they have persistency issue. We investigated *dbo:species* that shows issues.

Historical persistency is derived from persistency characteristic. It evaluates the percentage of persistency issues present over all KB releases. We argue that by persistency measure validation we also verified historical persistency results.

- lode:Event*:* From the extracted KB release on 2016-06-15, there are $2,182$ distinct entities of type *lode:Event*. However, in the 2016-09-09 release, that figure falls down to 689 distinct entities. We perform a comparison between the two releases to identify the missing entities. As a result we identified a total of 1911 entities missing in the newest release: this is an actual error. After a further investigation with the curators of the KB we found that this is due to an error in the reconciliation framework caused by a problem of overfitting. The error present in the 2016-09-09 release is a true positive identified by the Persistency measure.
- dbo:Species*:* We analyzed entity counts of type *dbo:Species* for the latest two releases of DBpedia (201510 and 201604). The counts are $305,378$ and $301,715$ respectively. We performed a comparison between the two releases to identify the missing entities; we found $12,791$ entities that are no more present in the latest release. We investigate in more detail the first six missing entities. For example, the entity *AIDS_II* [31] in 201510 was classified with type *dbo:Article* as well as *dbo:Species*. However, in 201604 it has been updated with a new type and the type *dbo:Species* was removed. There was clearly an error in the previous version that has been fixed in the latest, however, from the point of view of the latest release this is a false positive.

Consistency We computed the consistency measure values using the threshold $T = 100$. Properties with consistency = 0 were considered as potential quality issues. We considered the properties attached to entities typed *lode:Event* for the 2016-09-09 3cixty Nice KB. For the DBpedia KB, we considered the properties attached to the entities of type *foaf:Person* from the 201604 release.

- lode:Event *properties:* We found only 10 inconsistent properties. After a manual inspection of those properties we were unable to identify any actual error in the resources, so we classified all of the issues as false positives.
- foaf:Person *properties:* We extracted all the properties attached to entities of type *foaf:Person* and we identified 158 inconsistent properties. From the properties list, we inspected each of the property resources in detail. From the initial inspection, we observe that properties with low frequency contain actual consistency problems. For example, the property *dbo:Lake*[32] present in the class *foaf:Person* has a property frequency of 1. From further investigations, this page relates to *X. Henry Goodnough* an engineer and chief ad-

[31] http://dbpedia.org/page/AIDS_(computer_virus)
[32] http://dbpedia.org/ontology/Lake

vocate for the creation of the *Quabbin Reservoir project*. However, the property relates to a person definition. This indicates an error presents due to the wrong Wikipedia infobox extraction. From the manual validation, the precision of the identification of issues using the consistency measure accounts to 68% .

Completeness For the of 3cixty KB we analyzed the 2016-06-06 and 2016-09-09 releases, we observed the properties attached to *lode:Event* entities. For the entities typed as *foaf:Person* in 201510 and 201604 of the DBpedia KB releases, 50 have completeness issues. For manual validation we manually inspected whether they are real issues.

- lode:Event *properties:* From the analysis of the 2016-06-06 and 2016-09-09 releases of the 3cixty KB releases, we found eight properties showing completeness issues. Based on the eight *lode:Event* class properties, we investigated all entities and attached properties. We first investigated five instances for each property, manually inspecting 40 different entities. From the investigation we observed that those entities that are presents in 2016-06-06 are missing in 2016-09-09 that leads to a completeness issue. Entities are missing in the 2016-09-09 release due to an error of the reconciliation algorithm. Based on this manual investigation, the completeness measure generates an output that has a precision of 95%.
- foaf:Person *properties:* Based on the fifty properties from *foaf:Person* class identified by the completeness measure in the quantitative experiment, we investigated the subject of each property. We first checked five subjects for manual evaluation. For DBpedia, we checked a total of 250 entities. For example, we identified that the property *bnfId* has completeness issue. We extracted all the subjects for the releases of 201510 and 201610. In detail, the property *dbo:bnfId* for version 201610 has only 16 subjects and for version 201510 has 217 subjects. We performed a subject-wise comparison between these two releases to identify the missing subjects of the given property *dbo:bnfId* in the 201610 release. After a comparison between the two releases, we found 204 subjects missing in 201610 version of DBpedia. We perform a further manual investigation on subjects to verify the result. One of the results of the analysis is *John_Hartley _(academic)*[33] who is available in the 201510 release. However, it is not found in 201604 release of DBpedia. To further validate such an output, we checked the source Wikipedia page using *foaf:primaryTopic* about *John Hartley (academic)*[34]. In the Wikipedia page *BNF ID* is present as linked to external source. In DBpedia from 201510 version to 201604 version update, this subject has been removed for the property *dbo:bnfId*. This example clearly shows a completeness issue presents in the 201610 release of DBpedia for property *dbo:bnfId*. Based on the investigation over the property values, we compute our completeness measure has the precision of 94%.

6. Discussion

In this paper, we have proposed an approach to detect quality issues automatically leveraging four quality characteristics being derived by a temporal analysis over different KB releases. In this section, we first summarize our findings and then we discuss the threats that we have identified and have led the planning of future activities.

6.1. Temporal Analysis to Drive Quality Assessment

Similarly to Radulovic et al. [30], we will discuss our approach with respect to the following three criteria:

Conformance provides insights to what extent a quality framework and characteristics meet established standards. In fact, the four quality characteristics we have proposed derive from the ISO 25012 standard. A mapping between our quality characterises and ISO 25012 standard is reported in Table 2.

Applicability implies the practical aspects of the quality assessment approach. The main purpose of our approach is to automate the evaluation of a KB. We experimented with two different KBs and verified our hypothesis for both KBs. Our implementation follows a simple structure and it can scalable to KBs with a large number of entities and properties.

[33] http://dbpedia.org/page/John_Hartley_(academic)

[34] https://en.wikipedia.org/wiki/John_Hartley_(academic)

Performance We evaluated our quality assessment approach in terms of precision through manual evaluation. We evaluated precision based on completeness measure for the 3cixty and DBpedia KBs. The computed precision of completeness measure in our quality assessment approach is: *i)* 94% for the DBpedia KB *foaf:Person*-type entities and *ii)* 95% for the *lode:Event*-type entities of the 3cixty Nice KB. However, the capability of Consistency characteristics to detect quality issues varies significantly between the two case studies. We only identify Consistency issue in case of DBpedia and computed precision of 68% through manual evaluation for *foaf:Person*-type entities.

6.2. Frequency of Knowledge Base Changes

KBs can be classified according to application areas, schema changes, and frequency of data updates. The two KB we analyzed, namely 3cixty Nice and DBpedia, fall into two distinct categories: *i)* continuously changing KB with high frequency updates (daily updates), and *ii)* KB with low frequency updates (monthly or yearly updates).

i) KBs continuously grow because of an increase in the number of instances and predicates, while they preserve a fixed schema level (T-Box). These KBs are usually available via a public endpoint. For example the 3cixty Nice KB falls in this category. In fact, the overall ontology remains the same but new triples are added as effect of new information being generated and added to the KB. In our analysis, we collected batches of data at nearly fixed time intervals for 8 months.

ii) KBs grow at intervals since the changes can be observed only when a new release is deployed. DBpedia is a prime example of KBs with a history of releases. DBpedia consists of incremental versions of the same KB where instances and properties can be both added or removed and the schema is subjected to changes. In our approach we only considered subject changes in a KB over all the releases. In particular, we only considered those triples T from common classes $(c_1...c_i)$ or properties $(p_1...p_i)$ presented in all releases $(V_1....Vn)$ of the same KB.

6.3. Quality Assessment over Literal Values

We performed experimental analysis based on each quality characteristics. From the quantitative analysis, we identified properties with quality issues from consistency and completeness measures. We validated the observed results through manually investigating each properties value. From our investigation, we perceive that those properties that have quality issues may contain an error in literal values. We then further investigated our assumption in the case of DBpedia. We choose one random property of the *foaf:Person*-type entities. We finally examined the literal values to identify any error present.

From our quantitative analysis on the completeness characteristics of DBpedia, we detected the property *bnfId* [35] triggered a completeness issue. Only 16 resources in DBpedia 201604 version had such an issue, while 217 resources in 201510 version. We, therefore, further investigated the property *bnfId* in details on the 201604 release. We explored the property description that leads to Wikidata link[36] and examined how *BnF ID* is defined. It is an identifier for the subject issued by BNF (Bibliothèque nationale de France). It is formed by 8 digits followed by a check digit or letter. In Table 11, we present 6 subjects and objects of 207 *bnfId* property where each object follows the formatting structure. However, the literal value for subject *Quincy_Davis_(musician)*[37] contains a "/" between the digits "12148" and "cb16520477z", which does not follow standard formatting structure issued by BNF (Bibliothèque nationale de France). It clearly points to an error for the subject *Quincy_Davis_(musician)*.

From the initial inspection, we assume that it can be possible to identify an error in any literal value using our approach. However, to detect errors in literal values, we need to extend our quality assessment framework to inspect literal values computationally. We considered this extension of literal value analysis as a future research endeavour.

6.4. Knowledge Base Growth Assumption

On the basis of the continuous growth assumption [27], a further conjecture poses that the growth of the knowledge in a mature KB ought to be stable. From our analysis on the 3cixty Nice and DBpedia KB, we observed that variations in the knowledge base growth can affect quality issues. Furthermore, we argue that

[35] http://dbpedia.org/ontology/bnfId

[36] https://www.wikidata.org/wiki/Property:P268

[37] http://dbpedia.org/resource/Quincy_Davis_(musician)

Table 11
A sample of 6 subjects and objects of *bnfId* property (prefix dbp:http://dbpedia.org/resource)

Subject	Object
dbp:Tom_Morello	"14051227k"
dbp:David_Kherdian	"14812877"
dbp:Andrè_Trocmè	"cb12500614n"
dbp:Quincy _Davis_(musician)	"12148/cb16520477z"
dbp:Charles_S. _Belden	"cb140782417"
dbp:Julien_Durand _(politician)	"cb158043617"

quality issues can be identified through monitoring KB growth.

We can measure growth level of KB resources (instances) by measuring changes over the different releases. In particular, knowledge base growth can be measured by detecting the changes over KB releases utilizing trend analysis such as the use of simple linear regression. Based on the comparison between observed and predicted values, we can detect the trend in the KB resources, thus detecting anomalies over KB releases if the resources have a downward trend over the releases. Following, we derive KB growth measure regarding changes over time as well as experiments on the 3cixty Nice and DBpedia KB.

To measure KB growth, we applied linear regression analysis of entity counts over KB releases. In the regression analysis, we excluded the latest release to measure the normalized distance between an actual and a predicted value. In particular, in the linear regression we used entity count (y_i) as dependent variable and time period (t_i) as independent variable. Here, $n =$ *total number of KB releases* and $i = 1...n-1$ present as time period.

We start with a linear regression fitting the count measure of the class (C):

$$y = at + b$$

The residual can be defined as:

$$residual_i(C) = a \cdot t_i + b - count_i(C)$$

We define the normalized distance as:

$$ND(C) = \frac{residual_n(C)}{mean(|residual_i(C)|)}$$

Based on the normalized distance, we can measure KB growth of a class C as:

$$KBgrowth(C) = \begin{cases} 1 \; if ND(C) \geq 1 \\ 0 \; if ND(C) < 1 \end{cases}$$

the value is 1 if the normalized distance between actual value is higher than predicted value of type C , otherwise it is 0. In particular, if the KB growth measure has value of 1 then the KB may have unexpected growth with unwanted entities otherwise KB remains stable.

3cixty Nice case study The experimental data is reported in Table 4. We applied the linear regression over the eight releases for the *lode:Event*-type and *dul:Place*-type entities. We present the regression line in Figure 12a and 12b.

From the linear regression, the 3cixty Nice has a total of $n = 8$ releases where the 8^{th} predicted value for *lode:Event* $y'_{event_8} = 3511.548$ while the actual value=689. Similarly, for *dul:Place* $y'_{place_8} = 47941.57$ and the actual value=44968.

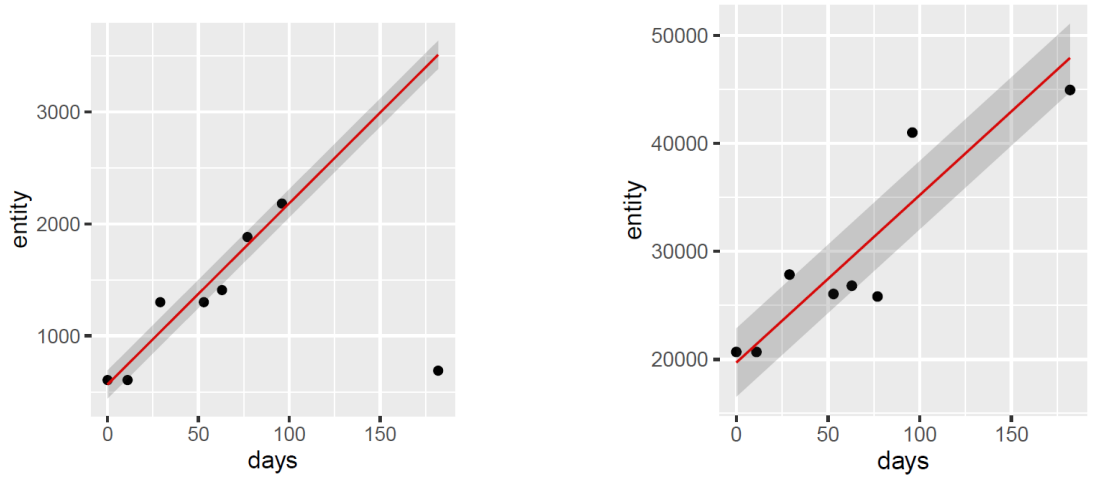
The residuals, e_{events_8}= $|689-3511.548| = 2822.545$ and e_{places_8}= $|44968 - 49741.57| = 2973.566$. The mean of the residuals, $e_{event_i} = 125.1784$ and $e_{place_i} = 3159.551$, where $i = 1...n-1$.

So the normalized distance for, 8^{th} *lode:Event* entity $ND_{event} = \frac{2822.545}{125.1784} = 22.54818$ and *dul:Place* entity $ND_{place} = \frac{2973.566}{3159.551} = 0.9411357$.

For the *lode:Event* class $ND_{events} \geq 1$ so the KB growth measure value = 1. However the for *dul:Place*, $ND_{places} < 1$ so the KB growth measure value =0 .

In case of the 3cixty Nice KB the *lode:Event* class clearly presents anomalies as the number of distinct entities drops significantly on the last release. In Figure 12a, the *lode:Event* class growth remains constant until it has errors in the last release. It has higher distance between actual and predicted value of *lode:Event*-type entity count. However, in the case of *dul:Place*-type, the actual entity count in the last release is near to the predicted value. We can assume that on the last release the 3cixty Nice KB has improved the quality of data generation matching the expected growth.

DBpedia Case study The experimental data is reported in Table 5. Based on the KB growth measure definition, we measured the normalized distance for each class (Table 12). We compared with the number of entities from the last release (201604) actual value and predicted value from linear regression to measure the normalized distance.



(a) lode:Event (b) dul:Place

Fig. 12. 3cixty two classes KB growth measure

From the results observed for *dbo:Artist*, *dbo:Film* and *dbo:MusicalWork*, the normalized distance is near the regression line with, and $ND < 1$. In Figure 13, we present the DBpedia 10 classes KB growth measure value and we can observe that there is no issue in the KB.

For instance while inspecting the different trends over the KB releases and calculating the normalized distance, we identified that *foaf:Person*-type last release (201604) entity count has a higher growth (over the expected). Such as *foaf:Person* has KB growth measure of 1 where normalized distance, $ND = 2.08$. From this measure we can implies that in *foaf:Person* there is persistency issue. We can imply that additions in a KB can also be an issue. It can include unwanted subjects or predicates.

We define this KB growth measure as *stability characteristic*. A simple interpretation of the stability of a KB is monitoring the dynamics of knowledge base changes. This measure could be useful to understand changes in a KB growth over time. Data curators can identify persistency issues in KB resources using growth analysis. However, a further exploration of the KB growth analysis is needed and we consider this as future research activity. In particular, we want to explore further *(i)* which factors are affecting KB growth and *(ii)* validating the stability measure.

6.5. Threats to validity

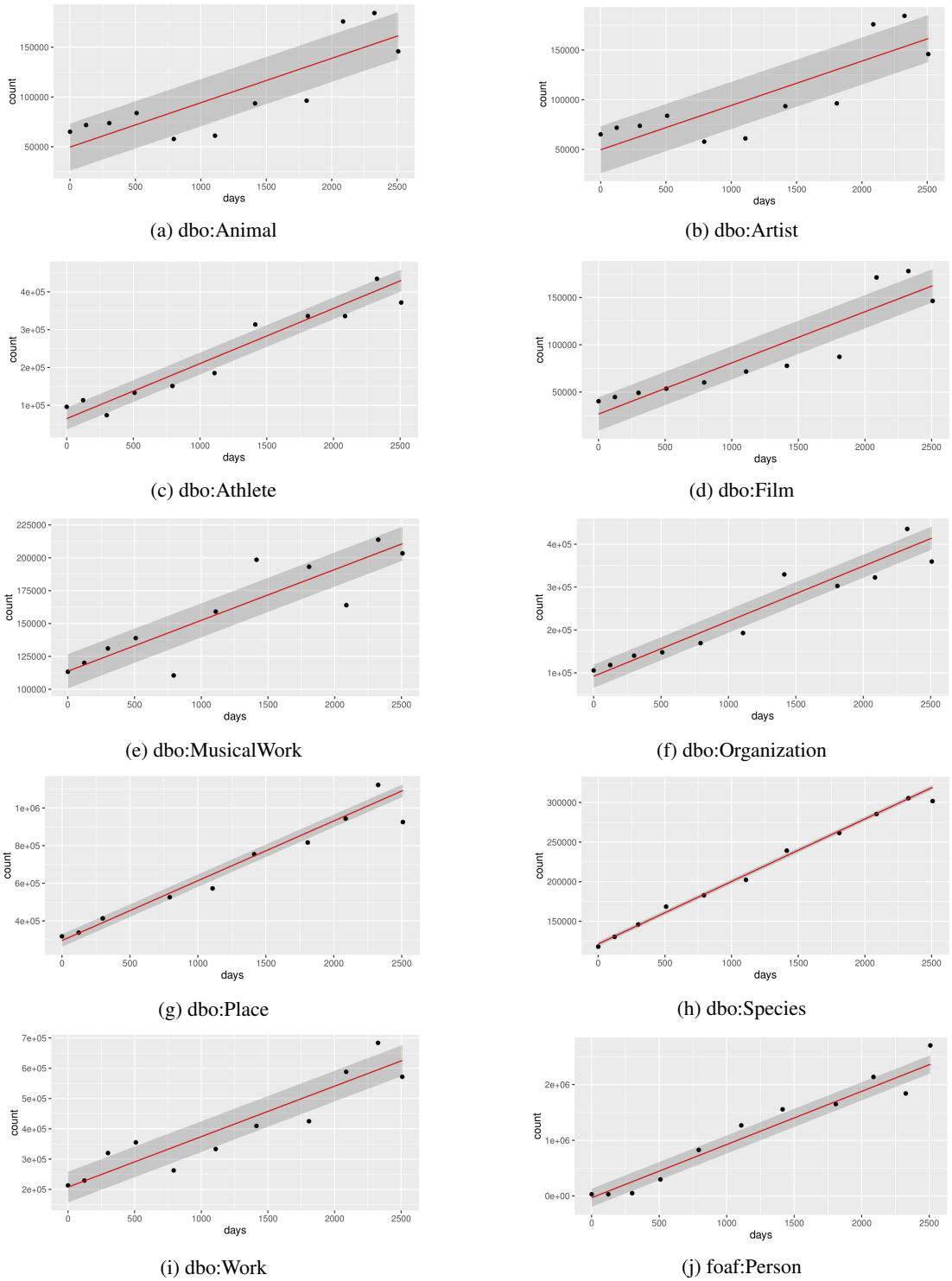
We have identified the following two main threats to validity.

Table 12
DBpedia 10 class Summary

Class	Normalized Distance(ND)	KB Growth measure
dbo:Animal	3.05	1
dbo:Artist	0.66	0
dbo:Athlete	2.03	1
dbo:Film	0.91	0
dbo:MucsicalWork	0.56	0
dbo:Organisation	2.02	1
dbo:Place	5.03	1
dbo:Species	5.87	1
dbo:Work	1.05	1
foaf:Person	2.08	1

First, as a basic measurement element we only considered aggregated summary data from statistical profiling such as frequency of properties in a class. We did not consider raw knowledge base differences among releases. This may lead to invalid results due to fact that we neglect individual additions or deletions of triples, focusing only on the count.

Let us consider the count of entities of a given type. Between two releases we may expect a set of entities E_- be removed and a set of entities E_+ entities to be added. If the numbers of entities in the two sets are equal ($|E_-| = |E_+|$), then no changes can be detected on the basis of count, even if the two set of entities are completely disjoint ($E_- \cap E_+ =$).



(a) dbo:Animal (b) dbo:Artist

(c) dbo:Athlete (d) dbo:Film

(e) dbo:MusicalWork (f) dbo:Organization

(g) dbo:Place (h) dbo:Species

(i) dbo:Work (j) foaf:Person

Fig. 13. DBpedia 10 classes KB growth measure

In order to be able to detect actual differences we would need to store two releases of a KB in a single graph in order to compute the set differences re-

quired to detect such changes. Alternatively we would have to extract the lists of all entities from two distinct endpoints and perform the set difference externally. Both solutions present several drawbacks from a technical point of view. Furthermore, regardless of the technical details, the set difference operation is, computationally-wise, extremely expensive.

Second, in this study we used 100 as threshold value for the consistency measure. We chose 100 since from the empirical analysis at property level it allowed to maximize the precision of the approach. However, the lack of an automatic adaptation and computation depending on the knowledge base can be considered as a threat to validity that we will consider as a future work.

7. Conclusion and Future Work

The main motivation for the work presented in this paper is rooted in the concepts of Linked data dynamics[38] on the one side and knowledge base quality on the other side. Knowledge about Linked Data dynamics is essential for a broad range of applications such as effective caching, link maintenance, and versioning [18].

However, less focus has been given toward understanding knowledge base resource changes over time to detect anomalies over various releases. In particular, we explored the idea of monitoring KB changes as the the premise of this work. We assumed that temporal analysis of the changes observed in different KB releases may lead to detecting issues and measuring quality. To verify our assumption, we proposed the measure of four quality characteristics, based on temporal analysis. We designed a quality assessment approach by profiling quality issues using different Knowledge Base (KB) releases. In this paper, we proposed temporal based measures for the quality characteristics persistency, historical persistency, consistency, and completeness to extend the quality characteristics from the ISO 25012 standard. We defined a KB quality assessment approach that explores the KB changes over various releases of the same KB. The proposed approach is able to provide a quality problem report to KB curators.

To assess our approach, we performed an experimental analysis based on quantitative and qualitative procedures. The assessment was conducted on two knowledge bases, namely DBpedia and 3cixty Nice. For the quantitative analysis, we applied our quality characteristics over eight releases of the 3cixty Nice KB and 11 releases of the DBpedia KB. Furthermore, we applied a qualitative analysis technique to validate the effectiveness of the approach. We can summarize the main findings of our work as follows:

- The analysis of Persistency and Historical Persistency on the 3cixty KB shows that KB changes over the releases could lead to detecting missing values. Missing values could happen due to algorithm error as in the case of the 3cixty Nice KB *lode:Event* class. However, in the case of DBpedia Persistency isues do not always indicate actual errors. We observed that *dbo:Species* subjects with the wrong type in version 201510 fixed in version 201610.
- From the quantitative and qualitative analysis of consistency measure, we only identify issues in case of DBpedia KB. We did not find any consistency issue for 3cixty Nice KB. We observe that continuously changing KBs with high-frequency updates (daily updates) such as 3cixty Nice KB tends to remain stable in case of the consistency issue. On the other hand, KB with low-frequency updates (monthly or yearly updates) such as DBpedia KB tends to have inconsistency.
- We observed extremely good performances of the completeness characteristics for both 3cixty Nice and DBpedia KB. The Completeness measure was able to detect actual issues with very high precision – 95% for the 3cixty Nice KB *lode:Event* class and 94% for the DBpedia *foaf:Person* class.

The future research activities we envision as follow up of this research endeavour are as follow:

- We plan to expand our temporal analysis approach by analyzing other quality characteristics presented in literature such as Zaveri *et al.* [38]. Also, we intend to apply our approach to KBs in other domain to further verify our assumption;
- A threat to validity of the current approach is that we only applied comparison between KB releases based on summary statistics. For instance, a comparison based on entity count of a class between two KB releases. We did not consider changes of raw KB releases. We plan to study how we can dynamically adapt one to one comparison over the various raw KB releases;

[38] https://www.w3.org/wiki/DatasetDynamics

– From the initial experiments, we assume that it can be possible to identify an error in literal value using our approach. We want to extend our quality assessment approach to inspect literal values;
– We argue that quality issues can be identified through monitoring KB growth. This has led to conceptualize the Stability quality characteristics meant to identify anomalies in a KB. We plan to monitor various KB growth to validate this assumption.

References

[1] Ahmad Assaf, Raphaël Troncy, and Aline Senart. Roomba: An extensible framework to validate and build dataset profiles. In *European Semantic Web Conference*, pages 325–339. Springer, 2015.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] Sören Auer, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Introduction to linked data and its lifecycle on the web. In *Proceedings of the 7th International Conference on Reasoning Web: Semantic Technologies for the Web of Data*, RW'11, pages 1–75, Berlin, Heidelberg, 2011. Springer-Verlag.

[4] Piero A Bonatti, Aidan Hogan, Axel Polleres, and Luigi Sauro. Robust and scalable linked data reasoning incorporating provenance and trust annotations. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):165–201, 2011.

[5] Jeremy Debattista, SÖren Auer, and Christoph Lange. LuzzuâĂŤa methodology and framework for linked data quality assessment. *Journal of Data and Information Quality (JDIQ)*, 8(1):4, 2016.

[6] Suzanne M Embury, Binling Jin, Sandra Sampaio, and Iliada Eleftheriou. On the Feasibility of Crawling Linked Data Sets for Reusable Defect Corrections. In *Proceedings of the 1st Workshop on Linked Data Quality (LDQ2014)*, 2014.

[7] G. Rizzo et al. 3cixty@expo milano 2015: Enabling visitors to explore a smart city. In *14^{t}h International Semantic Web Conference (ISWC)*, 2015.

[8] Annika Flemming. Quality characteristics of linked data publishing datasources. *Master's thesis, Humboldt-Universität of Berlin*, 2010.

[9] Christian Fürber and Martin Hepp. SWIQA - A Semantic Web information quality assessment framework. In *Proceedings of the 19th European Conference on Information Systems (ECIS 2011)*, volume 15, page 19, 2011.

[10] Matthew Gamble and Carole Goble. Quality, trust, and utility of scientific data on the web: Towards a joint model. In *Proceedings of the 3rd international web science conference*, page 15. ACM, 2011.

[11] Yolanda Gil and Donovan Artz. Towards content trust of web resources. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(4):227–239, 2007.

[12] Jennifer Golbeck and Aaron Mannes. Using trust and provenance for content filtering on the semantic web. In *MTW*, 2006.

[13] Thomas Gottron and Christian Gottron. Perplexity of index models over evolving linked data. In *European Semantic Web Conference*, pages 161–175. Springer, 2014.

[14] Christophe Guéret, Paul Groth, Claus Stadler, and Jens Lehmann. Assessing Linked Data mappings using network measures. In *The Semantic Web: Research and Applications*, pages 87–102. Springer, 2012.

[15] Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker. An empirical survey of linked data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14:14–44, 2012.

[16] Antoine Isaac and Riccardo Albertoni. Data on the web best practices: Data quality vocabulary. W3C note, W3C, December 2016. https://www.w3.org/TR/2016/NOTE-vocab-dqv-20161215/.

[17] ISO/IEC. 25012:2008 – software engineering – software product quality requirements and evaluation (square) – data quality model. Technical report, ISO/IEC, 2008.

[18] Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan. Observing linked data dynamics. In *Extended Semantic Web Conference*, pages 213–227. Springer, 2013.

[19] Michel Klein, Dieter Fensel, Atanas Kiryakov, and Damyan Ognyanov. Ontology versioning and change detection on the web. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 197–212. Springer, 2002.

[20] Magnus Knuth, Dimitris Kontokostas, and Harald Sack. Linked Data Quality: Identifying and Tackling the Key Challenges. In *Proceedings of the 1st Workshop on Linked Data Quality (LDQ2014)*, 2014.

[21] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. Test-driven evaluation of Linked Data quality. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 747–758. ACM, 2014.

[22] Yuangui Lei, Victoria Uren, and Enrico Motta. A framework for evaluating semantic metadata. In *Proceedings of the 4th international conference on Knowledge capture*, pages 135–142. ACM, 2007.

[23] Pablo N Mendes, Hannes Mühleisen, and Christian Bizer. Sieve: Linked Data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123. ACM, 2012.

[24] Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. Loupe-An Online Tool for Inspecting Datasets in the Linked Data Cloud. In *Proceedings of the 14th International Semantic Web Conference Posters Demonstrations Track*, pages 1–4, 2015.

[25] Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. Collaborative Ontology Evolution and Data Quality - An Empirical Analysis. In *OWL: Experiences and Directions – Reasoner Evaluation: 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016, Revised Selected Papers*, pages 95–114, 2017.

[26] Nandana Mihindukulasooriya, Giuseppe Rizzo, Raphaël Troncy, Oscar Corcho, and Raúl Garcıa-Castro. A two-fold quality assurance approach for dynamic knowledge bases: The 3cixty use case. In *(ESWC'16), International Workshop on Completing and Debugging the Semantic Web*, 2016.

[27] Vicky Papavasileiou, Giorgos Flouris, Irini Fundulaki, Dimitris Kotzinos, and Vassilis Christophides. High-level change detection in rdf (s) kbs. *ACM Transactions on Database Systems (TODS)*, 38(1):1, 2013.

[28] Nathalie Pernelle, Fatiha Saïs, Daniel Mercier, and Sujeeban Thuraisamy. Rdf data evolution: efficient detection and semantic representation of changes.

[29] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data quality assessment. *Commun. ACM*, 45(4):211–218, April 2002.

[30] Filip Radulovic, Raúl García-Castro, and Asunción Gómez-Pérez. Semquare - an extension of the square quality model for the evaluation of semantic technologies. *Comput. Stand. Interfaces*, 38(C):101–112, February 2015.

[31] Yannis Roussakis, Ioannis Chrysakis, Kostas Stefanidis, Giorgos Flouris, and Yannis Stavrakas. A flexible framework for understanding the dynamics of evolving rdf datasets. In *International Semantic Web Conference*, pages 495–512. Springer, 2015.

[32] Tong Ruan, Xu Dong, H Wang, and Y Li. Kbmetrics-a multipurpose tool for measuring quality of linked open data sets. In *The 14th International Semantic Web Conference, Poster and Demo Session*, 2015.

[33] Anisa Rula, Matteo Palmonari, and Andrea Maurino. Capturing the age of linked open data: Towards a dataset-independent framework. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 218–225. IEEE, 2012.

[34] Saeedeh Shekarpour and SD Katebi. Modeling and evaluation of trust with an extension in semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):26–36, 2010.

[35] Giri Kumar Tayi and Donald P Ballou. Examining data quality. *Communications of the ACM*, 41(2):54–57, 1998.

[36] Jürgen Umbrich, Stefan Decker, Michael Hausenblas, Axel Polleres, and Aidan Hogan. Towards dataset dynamics: Change frequency of linked open data sources. 2010.

[37] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[38] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality Assessment for linked Data: A Survey. *Semantic Web*, 7(1):63–93, 2016.