A Quality Assessment Approach for Evolving Knowledge Bases

Semantic Web 0 (2018) 1–36
IOS Press

A Quality Assessment Approach for Evolving Knowledge Bases

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. Assessing the quality of a Knowledge Base (KB) is a complex task as it often means measuring the quality of structured information, ontologies and vocabularies, and queryable endpoints. Popular knowledge bases such as DBpedia, YAGO2, and Wikidata have chosen the RDF data model to represent their data due to its capabilities for semantically rich knowledge representation. Despite its advantages, there are challenges in using RDF data model, for example, data quality assessment and validation. In this paper, we present a novel knowledge base quality assessment approach that relies on evolution analysis. The proposed approach uses data profiling on consecutive knowledge base releases to compute quality measures that allow detecting quality issues. Our quality characteristics are based on the KB evolution analysis and we used high-level change detection for measurement functions. In particular, we propose four quality characteristics: Persistency, Historical Persistency, Consistency, and Completeness. Persistency and historical persistency measures concern the degree of changes and lifespan of any entity type. Consistency and completeness measures identify properties with incomplete information and contradictory facts. 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. The capability of Persistency and Consistency characteristics to detect quality issues varies significantly between the two case studies. Persistency measure gives observational results for evolving KBs. It is highly effective in case of KB with periodic updates such as 3cixty KB. The Completeness characteristic is extremely effective and was able to achieve 95% precision in error detection for both use cases. The measures are based on simple statistical operations that make the solution both flexible and scalable.

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

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) [2] cloud hosting several Knowledge Bases (KBs) making available billions of RDF triples from different domains such as Geogra-

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

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

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

phy, Government, Life Sciences, Media, Publication, Social Networking, and User generated data.

Such KBs evolve over time: their data (instances) and schemas can be updated, extended, revised and refactored [15]. 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 example, the DBpedia KB [2] has been already available for a long time, with various versions that have been released periodically. Along with each release, DBpedia proposed changes at both instance and schema level. The changes at the schema level involve classes, properties, axioms, and mappings to other ontologies [28]. Usually, instance-level changes include resources typing, property values, or identify links between resources.

In this context, KB evolution is important for a wide range of applications: effective caching, link maintenance, and versioning [20]. However, unlike in more controlled types of knowledge bases, the evolution of KBs exposed in the LOD cloud is usually unrestrained, what may cause data to suffer from a variety of quality issues, both at a semantic (contradiction) and at a pragmatic level (ambiguity, inaccuracies). This situation clearly affects negatively data stakeholders – consumers, curators, etc. –. Therefore, ensuring the quality of the data of a knowledge base that evolves over time is vital. Since data is derived from autonomous, evolving, and increasingly large data providers, it is impractical to do manual data curation, and at the same time it is very challenging to do continuous automatic assessment of data quality.

Data quality, in general, relates to the perception of the "fitness for use" in a given context [43]. One of the common preliminary task for data quality assessment is to perform a detailed data analysis. Data profiling is one of the most widely used techniques for data analysis [32]. Data profiling is the process of examining data to collect statistics and provide relevant metadata [30]. Based on data profiling we can thoroughly examine and understand each KB, its structure, and its properties before usage. In this context, monitoring KB evolution using data profiling can help to identify quality issues.

The key concept behind this work is based on the work from Papavasileiou *et al.* [33] where they present a KB evolution study based on low-level and high-level changes. The authors considered low-level changes as the essential building block for high-level changes, since they are more fine-grained. High-level changes are more schema-specific and dependent on the semantics of data. More specifically, KB evolution can be analyzed using fine-grained "change" detection at low-level or using "dynamics" of a dataset at high-level. Fine-grained changes of KB sources are analyzed with regard to their sets of triples, set of entities, or schema signatures [6,31]. For example, fine-grained analysis at the triple level between two snapshots of a KB can detect which triples from the previous snapshots have been preserved in the later snapshots. Moreover, it can detect which triples have been deleted, or which ones have been added.

On the other hand, the dynamic feature of a dataset give insights into how it behaves and evolves over a certain period [31]. Ellefi *et al.* [7] explored the dynamic features considering the use cases presented by Käfer *et al.* [20]. *KB evolution analysis* using dynamic feature help to understand the changes applied to an entire KB or parts of it. It has multiple dimension regarding the dataset update behavior, such as frequency of change, changes pattern, changes impact and causes of change. More specifically, using dynamicity of a dataset, we can capture those changes that happen often; or changes that the curator wants to highlight because they are useful or interesting for a specific domain or application; or changes that indicate an abnormal situation or type of evolution [33,31].

Based on the high-level change detection, we aim to analyze quality issues in any knowledge base. The main hypothesis that has guided our investigation is: *Dynamic features from data profiling can help to identify quality issues.*

In this paper, we address the challenges of quality measurements for evolving KB using dynamic features from data profiling. We propose a KB quality assessment approach using quality measures that are computed using KB evolution analysis. We divide this research goal into three research questions:

RQ1: *Which dynamic features can be used to assess KB quality characteristics?*
We propose evolution-based measures 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 evolution-based quality characteristics?*
We propose an approach that profiles different releases of the same KB and 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 different KBs.

Traditional data quality is a largely investigated research field, and a large number of quality characteristics and measures is available. To identify a set of consistent quality characteristics using KB evolution analysis, in our approach we explored the guidelines from two data quality standards, namely ISO/IEC 25024 [19] and W3C DQV [18]. We also explored the comprehensive survey presented by Zaveri *et al.* [47] on linked open data quality. Concerning the KBs evolution, we explored dynamic features on the class level and the property level. We thus defined four evolution-based quality characteristics based on dynamic features. We use basic statistics (i.e., counts, and diffs) over entities from various KB releases to measure the quality characteristics. More specifically, we compute the entity count and instance count of properties for a given entity type. Measurement functions are built using entity count and the amount of changes between pairs of KB releases. We presented an experimental analysis that is based on quantitative and qualitative approaches. We performed manual validation for qualitative analysis to compute precision by examining the results from the 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 a quality assessment method for analyzing quality issues using dynamic features over different KB releases;
- We report about the experimentation of this approach on two KBs: DBpedia [2](encyclopedic data) and 3cixty [9] (contextual tourist and cultural data).

This paper is organized as follows: Section 2, presents motivational examples that demonstrate various important aspects of our quality assessment approach. In Section 3, we present the related work focusing on Linked data dynamics and quality measurement of linked data. Section 4 contains the definition of the proposed evolution-based quality characteristics and measurement functions. Section 5 describes our approach that relies on the KB evolution analysis and generates automatic measures concerning the quality of a KB. In Section 6, we present our empirical evaluation conducted on two different KBs, namely DBpedia and 3cixty Nice KB. Section 7 discusses the initial hypothesis, research questions and insights gathered from the experimentation. We conclude in Section 8, by summarizing the main findings and outlining future research activities.

2. Background and Motivations

Resource Description Framework (RDF)[3] is a graph-based data model which is the *de facto* standard in Semantic Web and Linked Data applications. RDF graphs can capture and represent domain information in a semantically rich manner using ontologies. An RDF KB is a well-defined RDF dataset that consists of RDF statements (triples) of the form (*subject*, *predicate*, *object*). RDF Schema (RDFS)[4] provides a data-modelling vocabulary for RDF data.

In our approach we used two KBs namely, 3cixty Nice KB and DBpedia KB. Here we report a few common prefixes used over the paper:

- DBpedia ontology URL[5] prefix: *dbo*;
- DBpedia resource URL[6] prefix: *dbr*;
- FOAF Vocabulary Specification URL[7] prefix: *foaf*;
- Wikipedia URL[8] prefix: *wikipedia-en*;
- 3cixty Nice event type URL[9] prefix: *lode*;
- 3cixty Nice place type URL[10] prefix:*dul*.

RDF has proven to be a good model for data integration, and there are several applications using RDF either for data storage or as an interoperability layer [33]. One of the drawbacks of RDF data model is the unavailability of explicit schema information that precisely defines the types of entities and their properties [31]. Furthermore, datasets in a KB are often inconsistent and lack metadata information. The main reason for this problem is that data have been extracted from unstructured datasets and their schema usually evolves. Within this context, our work explores two main areas: (1) evolution of resources and (2) impact of erroneous removal of resources in a KB. In particular, in our approach the main use case for exploring KB evolution from data is quality assessment.

[3] https://www.w3.org/RDF
[4] https://www.w3.org/TR/rdf-schema/
[5] http://dbpedia.org/ontology/
[6] http://dbpedia.org/resource/
[7] http://xmlns.com/foaf/0.1/
[8] https://en.wikipedia.org/wiki/
[9] http://linkedevents.org/ontology
[10] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

Taking into consideration KB evolution analysis, performing a fine grained analysis based on low-level changes means substantial data processing challenges. On the other hand, a coarse grained analysis using high-level changes can help to obtain an approximate indication of the quality a data curator can expect.

In general, low-level changes are easy to define and have several interesting properties [33]. Low-level change detection compares the current with the previous dataset version and returns the delta containing the added or deleted entities. For example, two DBpedia versions – 201510 and 201604 – have the property *dbo:areaTotal* in the domain of *dbo:Place*. Low-level changes can help to detect added or deleted instances for *dbo:Place* entity type. One of the main requirements for quality assessment would be to identify the completeness of *dbo: Place* entity type with each KB releases. Low-level changes can help only to detect missing entities with each KB release. Such as those entities missing in the 201604 version (e.g. dbr:A_Rúa, dbr:Sandiás,dbr:Coles_Qurense). Furthermore, these instances are auto-generated from Wikipedia Infobox keys. We track the Wikipedia page from which DBpedia statements were extracted. These instances are present in the Wikipedia Infobox as Keys but missing in the DBpedia 201604 release. Thus, for a large volume of the dataset, it is a tedious, time-consuming, and error-prone task to generate such quality assessment manually.

The representation of changes at low-level leads to syntactic and semantic deltas [45] from which it is more difficult to get insights to complex changes or changes intended by a human user. On the other hand, high-level changes can capture the changes that indicate an abnormal situation and generates results that are intuitive enough for the human user. High-level changes from the data can be detected using statistical profiling. For example, total entity count of *dbo:Place* type for two DBpedia versions – 201510 and 201604 – is 1,122,785 and 925,383 where the entity count of 201604 is lower than 201510. This could indicate an imbalance in the data extraction process without fine grain analysis. However, high-level changes require fixed set of requirements to understand underlying changes happen in the dataset. For example, assuming that the schema of a KB remains unchanged, a set of low-level changes from data correspond to one high-level change.

In our work, we analyze high-level changes to identify quality issues for evolving KBs. We defined a quality assessment method, which, given two entity type versions computes their difference and then based on detected changes identifies potential quality issues. In the ISO/IEC 25012 standard [19] define data quality as the degree to which a set of characteristics of data fulfills requirements. Such characteristics include completeness, accuracy or consistency of the data. Each of the quality characteristics identifies a specific set of quality issues. A data quality issue is a set of anomalies that can affect the knowledge base exploitation and any application usage. Such as under the completeness characteristics we can find problems regarding missing information. In this paper, we focused on three main quality issues of a knowledge base: *(i)* Lack of consistency, *(ii)* Lack of completeness, and *(iii)* Lack of persistency.

Lack of consistency when a KB is inconsistent with the reality it represents. In particular, inconsistency relates to the presence of unexpected properties.

As an example, let us consider DBpedia version 201510 where we can find the resource of type *foaf:Person dbpedia:X. Henry Goodnough* that represent an entity. While we find (as expected) a *dbo:birthDate* property for the entity, we unexpectedly find the property *dbo:Infrastructure/length*. This is a clear inconsistency: in fact, if we look at the ontology, we can check that the latter property can be used for a resource of type *dbo:Infrastructure*, not for a person.


X. Henry Goodnough

From Wikipedia, the free encyclopedia

X. Henry Goodnough, (1860–1935), engine

Goodnough Dike

Goodnough Dike the wet side

Official name	Goodnough Dike
Location	Ware
Coordinates	42°17′51″N 72°17′56″W
Construction began	1933
Opening date	1938
Operator(s)	MWRA
Dam and spillways	
Impounds	Beaver Brook
Height	264 ft (80.47 m)
Length	2,140 ft (652.3 m)
Width (base)	878 ft (267.61 m)
Reservoir	
Creates	Quabbin Reservoir

Figure 1. Example of inconsistent Wikipedia data.

To better understand where the problem lies, we need to look at the corresponding Wikipedia page wikipedia-en:X._Henry_Goodnough 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, considering DBpedia version 201604, for the resources of type *foaf:Person* there are 1035 distinct predicates, among which 142 occur only once. Such anomalous predicates suggests the presence of consistency issues that can be located either in the original data source or – i.e. Wikipedia for this case – or in the lack of filtering in the data extraction procedure.

Lack of completeness relates to the resources or properties missing from a knowledge base. This happens when information is missing from one version of the KB because it has been removed at given point during KB's evolution[11]. In general, causes of completeness issues are linked to errors in the data extraction pipeline. Such as missing instances in a KB that are auto generated from data sources. As an example, let us consider a DBpedia resource *dbpedia:Abdul_Ahad_Mohma* of type *dbo:Person/Astronauts*. When looking at the source Wikipedia page *wikipedia-en:Abdul_Ahad_Mohman*, we observe that the infobox shown in Figure 2 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.

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. That 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

Figure 2. Example of incomplete Wikipedia data.

dbo:Astronaut/TimeInSpace predicate over 634 astronaut resources (66%), while in the previous version they were 465 out of 650 astronauts (72%). Such a significant variation suggests the presence of a major problem in the data extraction procedure applied to the original source, i.e. Wikipedia.

Lack of persistency relates to unwanted removal of persistent 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 the label "Modéliser, piloter et valoriser les actifs des collectivités et d'un terrritoire grâce aux maquettes numériques: retours d'expériences et bonnes pratiques"[12]. This resource happened to be part of the 3cixty Nice KB since it has been created the first time, but in a release it got removed even though, according to the experts curating the KB, it should not have been removed.

This issue can be spotted by looking at the total frequency of entities of a given resource type. For example, *lode:Event* type two releases – 2016-06-15 and 2016-09-09 – total entity count 2,182 and 689. In particular in the investigated example we have observed an (unexpected) drop of resources of the type *event* between the previous release dated as 2016-06-15 and the considered released from 2016-09-09. Such count drop actually indicates a problem in the processing and integration of the primary sources that feed the KB.

Such problems are generally complex to be traced manually because they require a per-resource check over different releases. When possible, a detailed, low-level and automated analysis is computationally expensive and might result into a huge number of fine-

[11] Of course it is possible the item was never present in the KB at any time during its evolution, though this kind of mistake is not detectable just by looking at the evolution of the KB.

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

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

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

Table 1
Features of the two Analysis type features

Analysis level	Detail	Volume	Stakeholder
Low-level	fine-grained	Large	Data end-user
High-level	coarse-grained	Small	Data Curator

grained issue notifications. Such amount of information might cause an information overload for the user of the notifications. However, provided they are filtered, such low-level notifications can be useful to KB end-users to assess the suitability for their purposes.

The proposed approach provides an assessment of the overall quality characteristic and is not aimed at pin pointing the individual issues in the KB but it aims to identify potential problems in the data processing pipeline. Such approach produces a smaller number of coarse-grained issue notifications that are directly manageable without any filtering and provide a useful feedback to data curators. Table 1 summarizes the features of the two types of analysis.

In Figure 4 we present an conceptual represent of our quality assessment procedure. We divide our quality assessment procedure into three steps:

(i) Requirements: When a data curator initiates a quality assessment procedure, he/she needs to select an entity type. Furthermore, it is essential to ensure that the selected entity is present in all KB releases to verify schema consistency.

(i) Coarse grain analysis: high-level changes help to identify more context dependent features such as dataset dynamicity, volume, the design decision. We used statistical profiling to detect high-level changes from dataset.

(iii) Fine grain analysis: in general, high-level changes, being coarse-grained, cannot capture all possible quality issues. However, it helps to identify common quality issues such as an error in data extraction and integration process. On the other hand, fine grained analysis helps to detect detailed changes. In our approach we propose coarse grained analysis using data profiling and evaluate our approach using fine grained analysis. We use manual validation for fine grained analysis.

3. Related Work

The research activities related to our approach fall into two main research areas: *(i)* Linked Data Dynamics, and *(ii)* Linked Data Quality Assessment.

3.1. Linked Data Dynamics

Various research endeavours focus on exploring dynamics in linked data on various use-cases.

Umbrich *et al.* [44] present a comparative analysis on LOD datasets dynamics. In particular, they analyzed entity dynamics using a labeled directed graph based on LOD, where a node is an entity and an entity is represented by a subject.

Pernelle *et al.* [36] present an approach that detects and semantically represents data changes in RDF datasets. Klein *et al.* [21] 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.

Käfer *et al.* [20] 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 goes on-line or offline, how often it changes as well as they further investigate domain-level trends. They explore the RDF content of the core documents across

Figure 4. The Quality Assessment Procedure proposed in this paper.

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.

Papavasileiou *et al.* [33] 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.

Gottron and Gottron [15] 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.

Ruan *et al.* [40] 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 perspectives. 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.

Nishioka *et al.* [31] present a clustering techniques over the dynamics of entities to determine common temporal patterns. The quality of the clustering is evaluated using entity features such as the entities' properties, RDF types, and pay-level domain. In addition, they investigated to what extend entities that share a feature value change together over time.

3.2. 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 [13] 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 [12] 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. *Util-*

ity dimension, which intends to assess whether data fits the purpose and satisfies users' need.

Shekarpour and Katebi [42] 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: 1. propagation of trust assessment from data source to triples, and 2. aggregation of all triple assessments.

Golbeck and Mannes [14] 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, and assessibility. Furber and Hepp [11] 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 [10] 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.* [17] focus their work in assessment of mainly errors, noise and modeling issues. Lei *et al.* [24] 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.* [41] 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.* [16] 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.* [26] 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.* [23] 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.* [22] 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.* [8] 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.* [47] present a comprehensive systematic review of data quality assessment methodologies applied to LOD. They have extracted 26 quality dimensions and a total of 110 objective and subjective quality indicators. They organized linked data quality dimensions into following categories, 1. Contextual dimensions; 2. Trust dimensions; 3. Intrinsic dimensions; 4. Accessibility dimensions; 5. Representational dimensions; 6. Dataset dynamicity. They explored dataset dynamicity features based on three dimensions: 1. Currency (speed of information update regarding information changes); 2. Volatility (length of time which the data remains valid); 3. Timeliness (information is available in time to be useful). However, they didn't considered the evolution of KB changes and aspects of temporal analysis.

Ellefi *et al.* [7] present a comprehensive overview of the RDF dataset profiling feature, methods, tools, and vocabularies. They present dataset profiling in a taxonomy and illustrate the links between the dataset profiling and feature extraction approaches. They organized dataset profiling features into seven top-level categories: 1. General; 2. Qualitative; 3. Provenance; 4. Links; 5. Licensing; 6. Statistical. 7. Dynamics. In the qualitative features, they explored the data quality perspectives and presented four categories: 1. Trust (data trustworthiness); 2. Accessibility (process of accessing data); 3. Representativity (analyze data quality issues); 4. Context/Task Specificity (data quality analysis with respect to a specific tasks). We used the qualitative features to summarize the aforementioned linked data quality assessment studies and presented in Table 2.

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, which is instead the main contribution of our approach.

4. Quality Characteristics and Evolution Analysis

The definition of our quality characteristics started with the exploration of two data quality standard reference frameworks: ISO/IEC 25012 [19] and W3C DQV [18]. ISO/IEC 25012 [19] defines a general data quality model for data retained in 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[13]. It covers the quality of the data, how frequently is it updated, whether it accepts user corrections, and persistence commitments.

Besides, to further compare our proposed quality characteristics[14] we explored the foundational work on the linked data quality by Zaveri *et al.* [47]. They surveyed existing literature and identified a total of 26 different data quality dimensions (criteria) applicable to linked data quality assessment.

Since the measurement terminology suggested in these two standards differs, we briefly summarize the one adopted in this paper and the relative mappings in Table 3.

4.1. Evolution Analysis

Large Knowledge Bases (KBs) are often maintained by communities that act as curators to ensure their

[13] `https://www.w3.org/TR/vocab-dcat/`

[14] In our work we will identify the quality aspects using the term quality characteristics from ISO-25012 [19] that corresponds to the term quality dimension from DQV [18].

Table 2
Summary of Linked Data Quality Assessment Approaches

Paper	**Degree of Automation**	**Dataset Feature**	**Goal**
Gil and Arts [13]	Semi-Automatic	Trust	Focus their work on the concept of reputation (trust) of web resources
Gamble and Goble [12]	Semi-Automatic	Trust	Focus on evaluating trust of Linked Data datasets.
Shekarpour and Katebi [42]	Semi-Automatic	Trust	Focus on assessment of trust of a data source
Golbeck and Mannes [14]	Semi-Automatic	Trust	Focus on trust in networks and their approach is based on the interchange of trust, provenance, and annotations.
Bonatti *et al.* [4]	Semi-Automatic	Trust	Focus on data trust based on annotations such as Blacklisting,Authoritativeness and Linking
Furber and Hepp [11]	Semi-Automatic	Representativity	Focus on the assessment of accuracy, which includes both syntactic and semantic accuracy, timeliness, completeness, and uniqueness.
Flemming [10]	Semi-Automatic	Accessibility	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.
Rula *et al.* [41]	Automatic	Context Specificity	Start from the premise of dynamicity of Linked Data and focus on assessment of timeliness in order to reduce errors related to outdated data.
Gueret *et al.* [16]	Automatic	Context Specificity	Define a set of network measures for the assessment of Linked Data mappings.
Mendes *et al.* [26]	Semi-Automatic	Representativity	Developed a framework for Linked Data quality assessment.
Knuth *et al.* [22]	Semi-Automatic	Qualitative	They outline validation which, in their opinion, has to be an integral part of Linked Data lifecycle.
Emburi *et al.* [8]	Automatic	Context Specificity	They developed a framework for automatic crawling the Linked Data datasets and improving dataset quality.
Assaf *et al.* [1]	Automatic	Representativity	They propose a framework that handles issues related to incomplete and inconsistent metadata quality.
Debattista *et al.* [5]	Automatic	Representativity	They propose a conceptual methodology for assessing Linked Datasets, proposing Luzzu, a framework for Linked Data Quality Assessment.

Table 3
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 elements for quantify the measurement method	Measure Element	-
Quality measurement results that that characterize a quality feature	Numerical Value	Observation
Set of operations having the object of determining a value of a measure	Measurement	Measurement

quality [46]. KBs naturally evolve in time due to several causes: *i)* resource representations and links that are created, updated, and removed; *ii)* the entire graph can change or disappear [39]. 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);
- Domain area: depending on the specific domain, updates can be minor or substantial. For instance, social data is likely to be subject to wide fluctuations than encyclopedic data, which are likely to undergo smaller knowledge increments;
- 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 building 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 [33] for more user-friendly change management.

In this context, we focus on the aspects of data profiling for KB evolution analysis. According to Ellefi *et al.* [7], the dynamic features used in our approach are the following:

- *Lifespan*: Knowledge bases contain information about different real-world objects or concepts commonly referred as entities. Lifespan measures the change patterns of a knowledge base. Change patterns help to understand the existence and kinds of categories due to updates or change behavior. Also, lifespan represents the time period when a certain entity is available.
- *Stability*: It helps to understand to what extent the degree of changes impacts the overall state of the knowledge base. In this account, the degree of changes helps to identify what are the causes that trigger changes as well as the propagation effects.
- *Update history*: It contains basic measurement elements regarding the knowledge base update behaviour such as frequency of changes. The frequency of changes measures the update frequency of a KB resource. For example, the instance count of an entity type for various versions.

4.2. Evolution-based Quality characteristics and Measures

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

Zaveri et al. [47] classified quality dimensions into four groups: *i)* intrinsic, those that are independent of the user's 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. The quality dimensions we propose fall into the groups of intrinsic and representational.

In the context of RDF data model our approach focuses on two different types of elements in a KB: classes and properties. At triple level we only explored subjects and predicates thus disregarding the objects either resources or literals. To measure if a certain data quality characteristic is fulfilled for a given KB, each characteristic is formalized and expressed in terms of a measure with a value in the range $[0..1]$.

4.2.1. Basic Measure Elements

The foundation of our approach is the change at the statistical level regarding the variation of absolute and relative frequency count of entities between pairs of KB versions.

In particular, we aim to detect variations of two basic statistical measures that can be evaluated with the most simple and computationally inexpensive 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 entities t of the type 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 properties, within a class C. We define the frequency of a property (in the scope of class C) as:

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

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 measurement element that can be used to build derived measures: the number of properties present for the entity type C in the release i of the KB.

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

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>.
}
```

In the remainder, we will use a subscript to indicate the release the measure refers to. The releases are numbered progressively as integers starting from 1 and, by convention, the most recent release is n. So, for instance, $count_{n-1}(\textit{foaf:Person})$ represents the count of resources typed with *foaf:Person* in the last but one release of the knowledge base under consideration.

4.2.2. Persistency

We define the Persistency characteristics as the degree to which erroneous removal of information from current version may impact stability of the resources. Ellefi *et al.* [7] present stability feature as an aggregation measure of the dataset dynamics. In this context, Persistency characteristics measures helps to understand stability feature. It provides insights into whether there are any missing resources in the last KB release.

An additional important feature to be considered when analyzing a 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 [44]. Persistency measures provide 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:

$$\textit{PersistencyClass}_i(C) = \begin{cases} 1 \text{ if } \textit{count}_i(C) \geq \textit{count}_{i-1}(C) \\ 0 \text{ if } \textit{count}_i(C) < \textit{count}_{i-1}(C) \end{cases}$$

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:

$$\textit{PersistencyKB}_i(KB) = \frac{\sum_{j=1}^{NC_i} \textit{PersistencyClass}_i(C_j)}{NC_i}$$

where NC_i is the number of classes analyzed where i is the release of the KB.

4.2.3. Historical Persistency

Historical persistency is a derived measurement function based on the persistency quality characteristic. It captures the whole lifespan of a KB with the goal of detecting quality issues, in several releases, for a specific entity-type [7]. It considers all entities pre-

sented in a KB and provides an overview of the whole KB. It also helps data curators to decide which knowledge base release can be used for future data management tasks.

The Historical Persistency measure is computed as the average of the pairwise persistency measures for all releases.

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

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

$$\text{H\_PersistencyKB}(KB) = \frac{\sum_{i=2}^{n} \text{H\_PersistencyClass}_i}{n-1}$$

4.2.4. Consistency

Consistency checks whether inconsistent facts are included in a KB. This quality characteristic relates to the Consistency quality characteristic defined in the ISO/IEC 25012 standard. The standard defines it as the "*degree to which data has attributes that are free from contradictions 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*" [19]. *Zaveri et al.* also explored the Consistency characteristics. In detail, a knowledge base is defined to be consistent if it does not contain conflicting or contradictory facts.

We assume that extremely rare predicates are potentially inconsistent, see e.g. the *dbo:Infrastructure/length* property discussed in the example presented in Section 2. We can evaluate the consistency of a predicate on the basis of the frequency distribution for an entity type.

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 or can defined on the basis of the count of the scope class. Paulheim and Bizer [35] explore the problem of consistency of RDF statements using the SDValidate approach. Their work is based on the observation that RDF statements with a frequent predicate/object combination are more likely to be correct than a small number of "outlier" statements with an infrequent predicate/object combination. We used a similar approach to derive our threshold value. Similarly to the SDValidate approach, we assume that properties with low relative frequency are more error-prone. In this account, in our threshold value analysis, we have explored the frequency distribution of properties to identify the threshold value. Furthermore, instead of one version, we considered multiple versions to assess a threshold value empirically.

We started our threshold value analysis by using a histogram of property frequencies distribution. From our initial observation, it is suitable to say that a good threshold value could be a point where there is a trend present in the distribution. Here the word trend should be interpreted as "the way things are heading," as it, e.g., a possible variation in the property frequency distribution. Concluding this reasoning, we come to the assumption that a good threshold point should be located at an extreme value in the first derivative of our histogram. To identify this changes in the histogram, we simply focus on the kernel density estimation (KDE) [34]. In general, it is a non-parametric way of the estimation of the density function of a univariate probability distribution[38].

In our approach, we use the local minimum of the KDE based on the property frequency distribution as a threshold value. However, in most cases, a priori knowledge must be applied to select the most appropriate threshold [25]. In this account, we chose various trend point such as 50, 100, 200, and 500 to maximize the precision of the qualitative analysis results. On the other hand, the number of properties varies with each KB release. Therefore, we also evaluated the last three releases of a KB to further validate our assumption. From our empirical analysis (Sec. 6.2.3) , we considered 100 as the threshold value by evaluating properties present in various KB releases that are optimized in the context of our qualitative analysis.

4.2.5. Completeness

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*" [19].

In Zaveri *et al.*, Completeness consists in the degree to which all required information is present in a

particular dataset. In terms of Linked Data, completeness comprises 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.

Evolution-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 2).

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:

$$\mathrm{NF}_i(p,C) = \frac{\mathrm{freq}_i(p,C)}{\mathrm{count}_i(C)}$$

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

$$\mathit{Completeness}_i(p,C) = \begin{cases} 1, & \mathrm{NF}_i(p,C) \geq \mathrm{NF}_{i-1}(p,C) \\ 0, & \mathrm{NF}_i(p,C) < \mathrm{NF}_{i-1}(p,C) \end{cases}$$

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

$$\mathit{Completeness}_i(C) = \frac{\sum_{k=1}^{\mathrm{NP}_i(C)} \mathit{Completeness}_i(p_k,C)}{\mathrm{NP}_i(C)}$$

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

5. Evolution-based Quality Assessment Approach

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, several versions of data life cycles exist with differences attributable to variation in practices across domains or communities [3]. Data quality life cycle generally includes the identification of quality requirements and relevant metrics, quality assessment, and quality improvement [5,29]. 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 continuous process. However, we explored the features of quality assessment based on KB evolution. Our reference Data Life Cycle is defined by the international standard ISO 25024 [19]. We extend the reference DLC to integrate a quality assessment phase along with the data collection, data integration, and external data acquisition phase. This phase ensures data quality for the data processing stage. The extended DLC is reported in Figure 5. The first step in building the quality assessment approach was to identify the quality characteristics.

Based on the quality characteristics presented in Section 4.2, we proposed a KB quality assessment approach. In particular, our evolution-based quality assessment approach computes statistical distributions of KB elements from different KB releases and detects anomalies based on evolution patterns. Figure 6 illustrates the workflow based on the quality assessment procedure we outlined in Section 2 and framed as a three-stage process: *(1)* input data (multiple releases of a knowledge base), *(2)* quality evaluation process, and *(3)* quality reporting. We implemented a prototype using the R statistical package that we share as open source in order to foster reproducibility of the experiments[15]. The stages are explained in detail below.

5.1. Input data

In our approach, we considered an entity type and history of KB releases as an input. The acquisition of KB releases can be performed by querying multiple SPARQL endpoints (assuming each release of the KB is accessible through a different endpoint) or by load-

[15] https://github.com/rifat963/KBQ

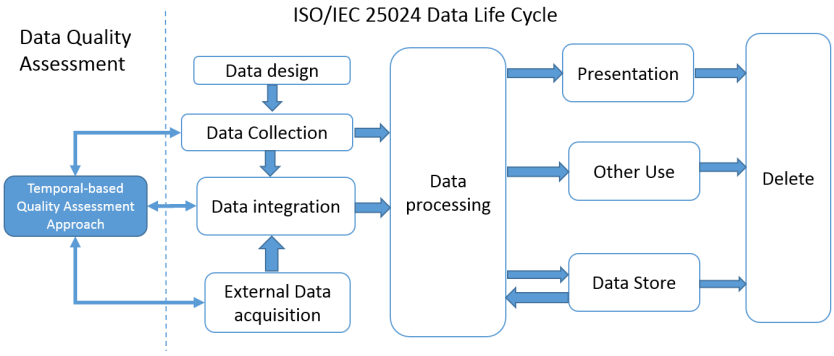


Figure 5. ISO/IEC 25024 Data Life Cycle [19] with proposed quality assessment approach.

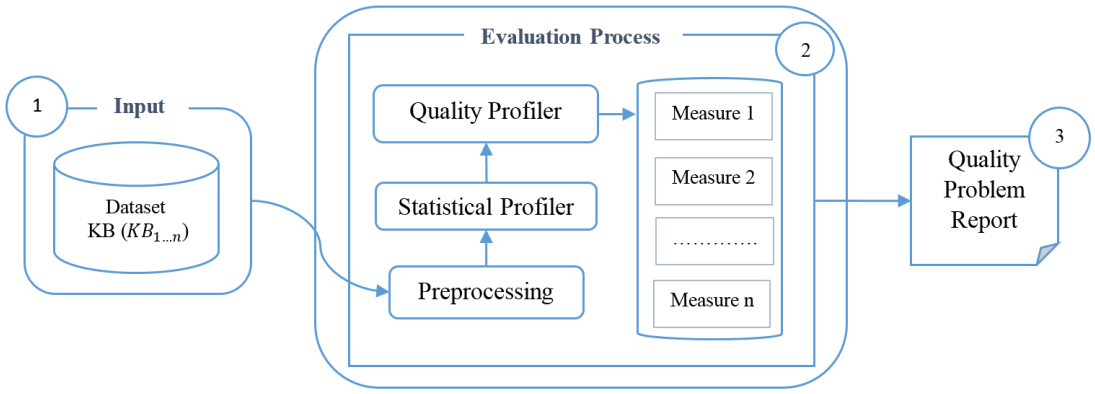


Figure 6. Workflow of the proposed Quality Assessment Approach.

ing data dumps. The stored dataset can be organized based on the entity type and KB release. We thereby build an intermediate data structure constituting an entity type and KB releases. We used this intermediate data structure as an input to the next step. Figure 7 reports the intermediary data structure that is used in the following stage.

In our implementation, we created a data extraction module that extends Loupe [27], 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. We used SPARQL endpoint as an input and save the results extracted from the SPARQL endpoints into the CSV files. We named each CSV file based on the knowledge base release and corresponding class name. In Figure 8, we illustrate the entity type 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 class *foaf:Person* and

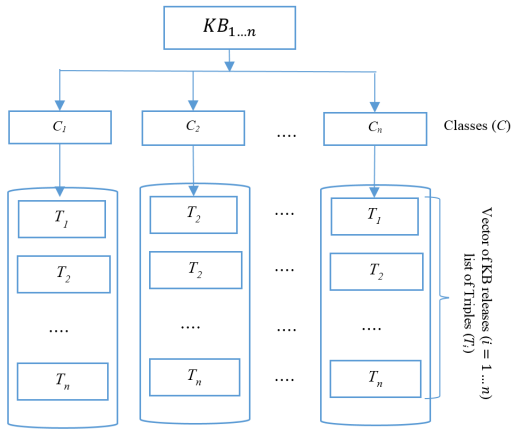


Figure 7. Intermediary data structure that is used as input for the Evaluation Process.

saved them into CSV files named with the names of the DBpedia releases.

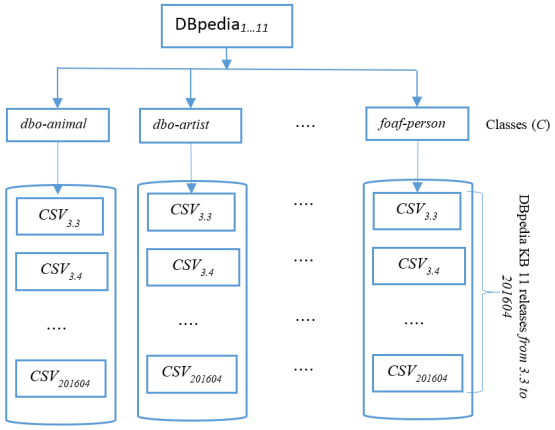


Figure 8. Structure of input module.

5.2. Quality Evaluation Process

We argue that anomalies can be identified using a combination of data profiling and statistical analysis techniques. We adopt a data-driven measurements of changes over time in different releases. The knowledge base quality is performed based on quality characteristics presented in Section 4.2. Firstly, the quality characteristics are evaluated by examining multiple KB releases; then, the result of quality assessment consists of quality information for each assessed knowledge base. This generates a quality problem report that can be as detailed as pinpointing specific issues at the level of individual triples. These issues can be traced back to a common problem and can be more easily identified starting from a high-level report.

The evaluation process includes the following three steps:

1. **Preprocessing:** In this component, we perform preprocessing operations over the intermediate data structure based on schema consistency checks. In particular, the goal of this component is to check if the chosen entity type is present in all the KB releases to verify schema consistency. This is essential to perform the schema consistency checks due to high-level changes that are more schema-specific and dependent on the semantics of data. More specifically, this component does the following tasks: *(i)* selection of only those entity types that are present in all KB releases; *(ii)* and for each entity type, selection of only those predicates present in that class. Furthermore, in our implementation, we have filtered those properties for an entity type in the intermediate data structure in case the instance count is 0 for all the KB releases.
2. **Statistical Profiler:** Then, in order to identify the dynamic feature of the sequence of KB releases, we compute the following key statistics using basic statistical operations:
   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 4.2.1. We compute change detection between two KB releases by observing the variation of key statistics. We divided 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.
3. **Quality Profiler:** Typically, 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 approach of data profiling together with quality measure to profile quality issues. We used the statistical profiler to analyze the KB releases. For analyzing the KB datasets, we used four quality characteristics presented in Section 4.2. Quality profiler includes descriptive as well as measure values based on the quality characteristics.
   Elaborating further, 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, 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 analyzed triples.

5.3. Quality Problem Report

We generate a quality report based on the quality assessment results. The reports contain quality mea-

sure computation results as well as summary statistics for each class. The 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. We implemented the quality problem report visualization using R markdown documents. R markdown documents are fully reproducible and easy to perform analyses that include graphs and tables. We presents an example of Quality problem report in the github repository[15].

6. Experimental Assessment

This section reports an experimental assessment of our approach that has been conducted on two KBs, namely DBpedia and 3cixty Nice. The analysis is based on the quality characteristics and measures described in Section 5, the measurement has been conducted by means of a prototype implementation of a tool as described in Section 5. Table 4 reports the interpretation criteria we adopted for each quality characteristic measure. We first present the experimental setting of the implementation then we report the results of both *i)* a quantitative and *ii)* a qualitative validation.

6.1. Experimental Settings

In our experiments, we selected two KBs according to three main criteria: *i)* popularity and representativeness in their domain: DBpedia for the encyclopedic domain, and 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 details:

- *DBpedia*[16] 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[17], 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[18], 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 [9], 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 [29].

We present a detailed summary of extracted datasets for each KB.

- *3cixty Nice*: We used public SPARQL endpoint of the 3cixty Nice KB in our data extraction module. As the schema in 3cixty KB remains unchanged, we used the same SPARQL endpoint for 8 different releases of 3cixty Nice KB. In particular, we considered eight different releases of the 3cixty Nice KB: from 2016-03-11 to 2016-09-09. We considered those instances having the *rdf:type*[19] of *lode:Event* and *dul:Place*. The distinct instance

[16] http://wiki.dbpedia.org

[17] https://www.wikipedia.org

[18] https://www.3cixty.com

[19] https://www.w3.org/1999/02/22-rdf-syntax-ns#type

Table 4
Verification conditions of the quality measures

Quality Characteristics	Measure	Interpretation
Persistency	Persistency measure values of 0 or 1.	The value of 1 implies no persistency issue present in the class. The value of 0 indicates persistency issues found in the class.
Historical Persistency	Percentage (%) of historical persistency	High % presents an estimation of fewer issues, and lower % entail more issues present in KB releases.
Completeness	List of properties with completeness measures weighted value of 0 or 1.	The value of 1 implies no completeness issue present in the property. The value of 0 indicates completeness issues found in the property.
	Percentage (%) of completeness	High % presents an estimation of fewer issues, and lower % entail more issues in KB release.
Consistency	List of properties with consistency measures value of 0 or 1.	The value of 1 implies no completeness issue present in the property. The value of 0 indicates completeness issues found in the property.

count for each class is presented in Table 5. The variation of *count* in the dataset and the observed history is presented in Figure 9. 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.

Table 5
3cixty Entity Count

Releases	*lode:Event*	*dul:Place*
2016-03-11	605	20,692
2016-03-22	605	20,692
2016-04-09	1,301	27,858
2016-05-03	1,301	26,066
2016-05-13	1,409	26,827
2016-05-27	1,883	25,828
2016-06-15	2,182	41,018
2016-09-09	689	44,968

– *DBpedia*: In our data extraction module, we directly used services provided by Loupe to access multiple DBpedia KB releases SPARQL endpoint to extract all triples for the selected ten classes. In the case of DBpedia, we considered ten classes: *dbo:Animal dbo:Artist*, *dbo:Athlete*, *dbo:Film*, *dbo:MusicalWork*, *dbo:Organisation*, *dbo:Place*, *dbo:Species*, *dbo:Work*, *foaf:Person*. The above entity types 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 6 presents the breakdown of frequency per class.

6.2. Quantitative Analysis

We applied our quantitative analysis approach based on the proposed quality characteristics. In particular, we analyzed the aforementioned selected 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 4, we present the interpretation criteria for each quality characteristic measure. We discuss in this section the quality characteristic analysis performed on each knowledge base.

6.2.1. Persistency

3cixty. Table 5 reports the entity count measure; in particular we highlight the latest two releases that are considered in computing Persistency according to the definition (Section 4.2). In the case of *lode:Event*-type instances, we can observe that $count_n = 689$ and $count_{n-1} = 2182$, where $n = 8$. Since we have $count_n < count_{n-1}$, the value of *Persis-*

Table 6
DBpedia 10 Classes entity count (all classes have *dbo:* prefix except the last one.

Version	Animal	Artist	Athlete	Film	MusicalWork	Organisation	Place	Species	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

tency(*lode:Event*)-type = 0. That indicate persistency issue present in the last KB release for the *lode:Event* class.

Similarly, concerning *dul:Place*-type instances, from the dataset we can see that $count_n = 44968$ is greater than $count_{n-1} = 41018$, therefore the value of *Persistency*(*dul:Place*) = 1. Thus, no 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 or 1. The 3cixty Nice KB percentage of Persistency (%)=$(\frac{\text{No. of classes with issues}}{\text{Total no. of classes}}) * 100 = (\frac{1}{2}) * 10 = 50\%$.

DBpedia. We compare the last two releases (201510, 201604) in terms of entity counts for ten classes; the two releases are highlighted in Table 6. The resulting Persistency measure values are reported in Table 7. For example, 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 is 0 since the entity counts in version 201604 are consistently lower than in 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*.

Discussion

According to the interpretation criteria reported in Table 4 we summarize our findings:

– In the case of 3cixty Nice KB, *lode:Event* class, *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 3cixty Nice KB. For both investigated types, the percentage of knowledge base Persistency is 50%, which triggers a warning concerning a potential persistency issue existing in the latest (2016-09-09) KB release.

Table 7
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%

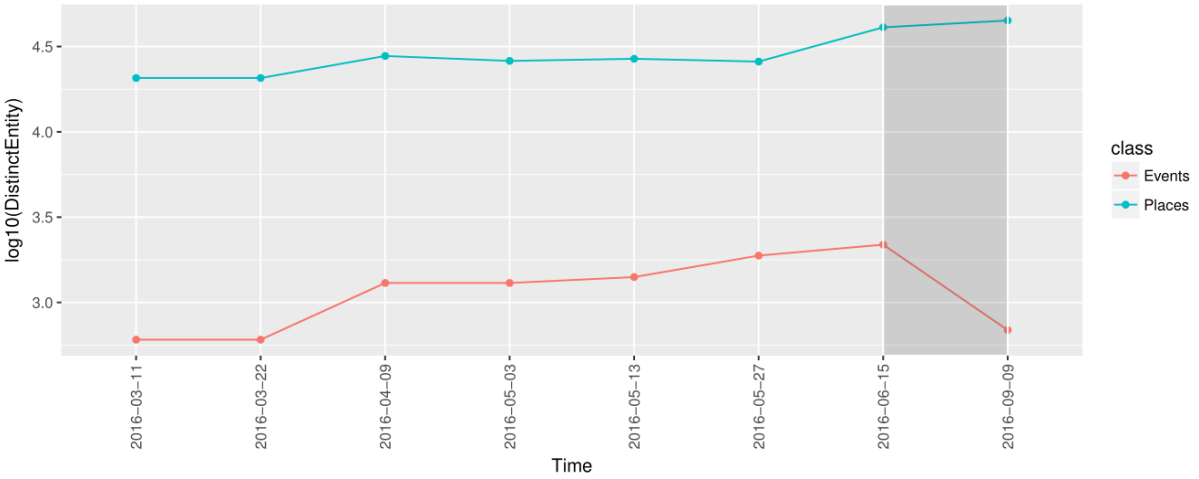


Figure 9. Variation of instances of 3cixty classes *lode:Event* and *dul:Place* over 8 releases.

– In the case of DBpedia KB, the analysis conducted using the persistency measure, only the *foaf:Person* class has persistency measure value of 1 indicating no issue. 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%.
– The Persistency measure is an observational measure. It only provides an overview of the KB degree of changes. It is effective in the case of rapid changes such as *lode:Event* class.

6.2.2. Historical Persistency

3cixty The variations of persistency measure are considered between the 2016-06-15 and the 2016-09-09 releases. The computation starts from the persistency measures presented in Table 5. For *lode:Event*-type entities the number of persistency variations with value of 1 is $= 6$. Therefore, concerning 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= $(\frac{5}{7}) * 100 = 71.42\%$.

DBpedia. Figure 10 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 7 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 $(\frac{5}{9}) * 100 = 55.55\%$.

Discussion

The Historical Persistency quality measure provides an overview of the different KB releases. It identifies those versions with persistency issues along the different KB releases. To recap:

– 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). Thus, overall historical persistency measure of *lode:Event* class higher than *dul:Place* class.
– In the case of DBpedia KB, 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%.
– Historical Persistency is mainly an observational measure and it gives insights on lifespan of a

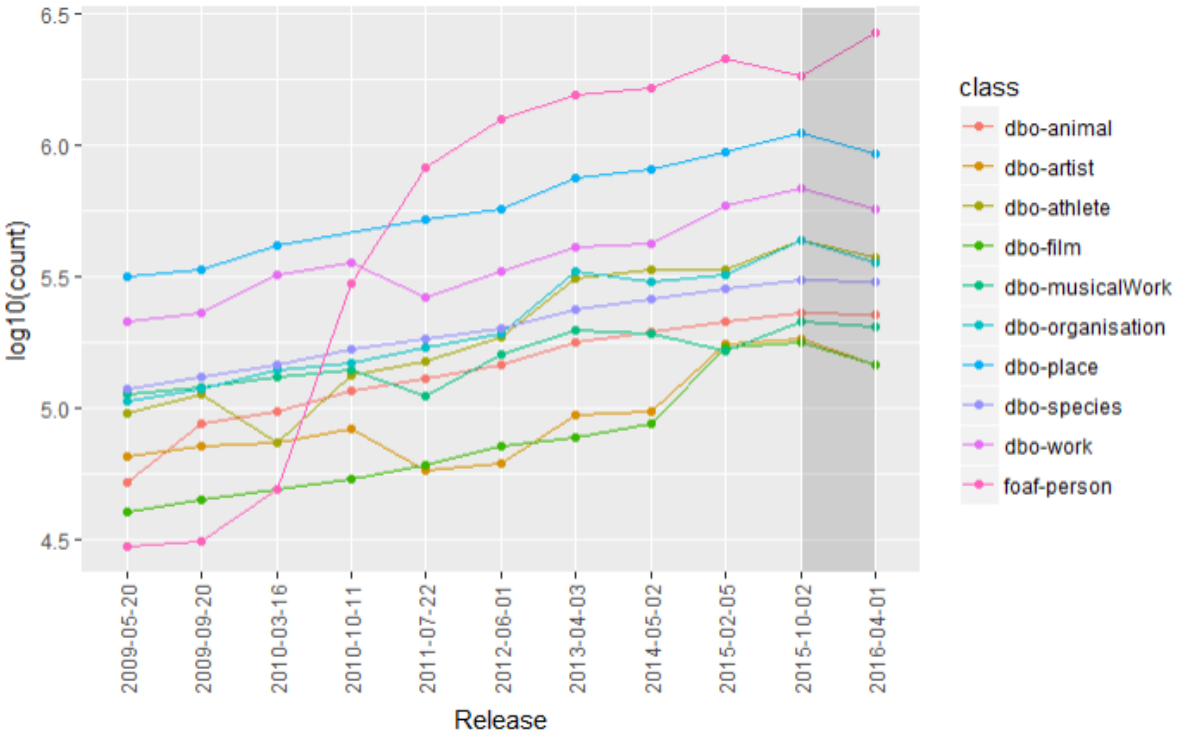


Figure 10. DBpedia 10 Classes instance variation over 11 releases.

KB. Using this measure value, a 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.

6.2.3. Consistency

Threshold Value. In this experimental analysis, we started by observing histogram of property frequencies distribution and kernel density estimation. For example, 3cixty Nice *lode:Event*-type releases (2016-05-27, 2016-06-15, 2016-06-09) frequency value of 150 has (12, 16, 10) properties, 100 has (12, 15, 10) properties and 50 has (2, 3, 2) properties. In this use case, we found a small number of properties with infrequent distribution. On the other hand, DBpedia KB *foaf:Person*-type frequency distribution for three releases (201504,201510,201604) with the threshold value of 200 has (178,177,167) properties, 100 has (164,164,158) properties, and 50 has (154,134,126). Figure 11 illustrates DBpedia *foaf:Person* class property frequencies distribution.

From the *foaf:Person* class kernel density estimation based on three releases, the average value of local minimum is 87.63. In this account, the threshold value of 50 is lower than the local minimum and has the lowest number of properties. On the other hand, the threshold value of 100 is near to the local minimum. Also, the threshold value of 100 has the maximum number of properties which is optimized for our qualitative analysis approach. Thus, we chose 100 since from the empirical analysis at property level it allowed to maximize the precision of the approach.

3cixty. we focus on the latest release (2016-09-09) of the 3cixty Nice KB. We analyzed *lode:Event*-type and *dul:Place*-type instances. Based on the threshold values of 100, we measured the consistency for *lode:Event* and *dul:Place*-type. From the *lode:Event*-type resources, by applying the consistency analysis, we found that 10 properties reported below the threshold. Similarly, for *dul:Place*-type resources we found that 12 properties below the threshold value.

DBpedia. Table 8 reports, for the DBpedia ten classes, the total number of properties, the *inconsistent* properties – i.e. those with consistency value = 0 –, and the consistent properties – consistency value = 1.

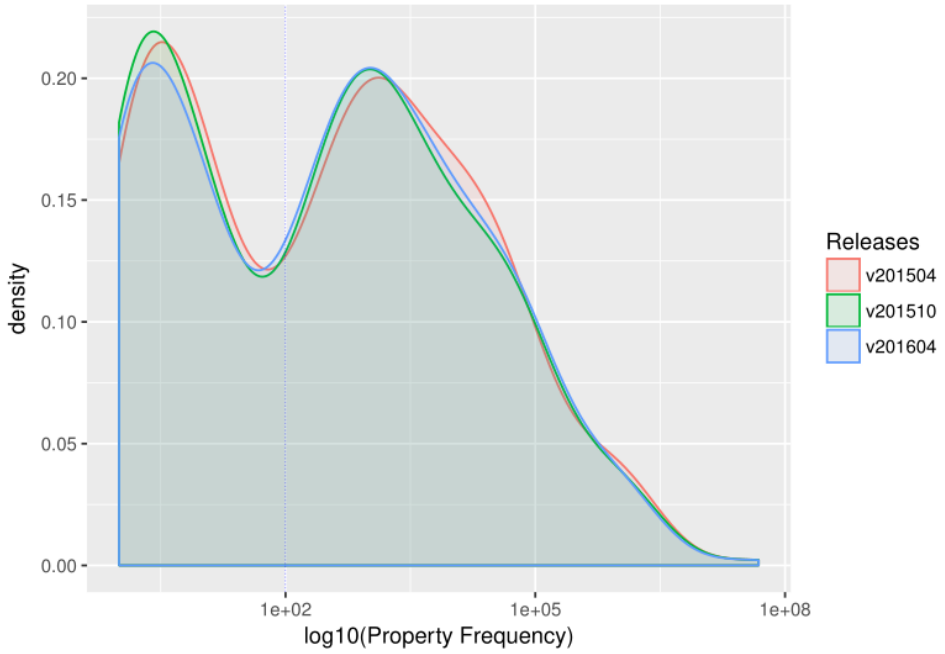


Figure 11. DBpedia *foaf:Person* class property frequencies distribution.

The values are based on the last release 201604. We measured the consistency by identifying those properties with the frequency lower than the threshold value $T = 100$. For example, *foaf:Person* has a total of 381 properties in the 201604 release. We found 158 inconsistent properties, i.e. properties whose frequency is lower than the threshold.

Discussion

The consistency measure is based on the assumption that properties with low relative frequency more error-prone and applicable to all KB releases. More specifically, we are interested in identifying properties with low relative frequency for an entity type. The main findings are:

- 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 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.
- In the last release (201604) of the DBpedia KB, we have identified consistent properties for 10 classes. Consistency measure results illustrated in Table 8. For example *foaf:Person* class has 158 inconsistent properties. We further investigate this measure for *foaf:Person* class through manual evaluation.

6.2.4. Completeness

3cixty. 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 4.2), for the *lode:Event*-type, the number of predicates in the last two releases $= 21$ and the number of predicates with completeness issues (value of 0) $= 8$. In Figure 12, 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*-type is $(\frac{13}{21}) * 100 = 62\%$. Similarly, for *dul:Place*-type, the number of predicates in the last two releases $= 28$ and the number of predicates with completeness issue (value of 0) $= 14$. In Figure 13, we present *dul:Place*-type completeness measure results of those properties with completeness issue (value of 0).

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

DBpedia. Table 9 illustrates the results of the completeness measure based on the latest two releases of DBpedia 201510 and 201604. This table reports the

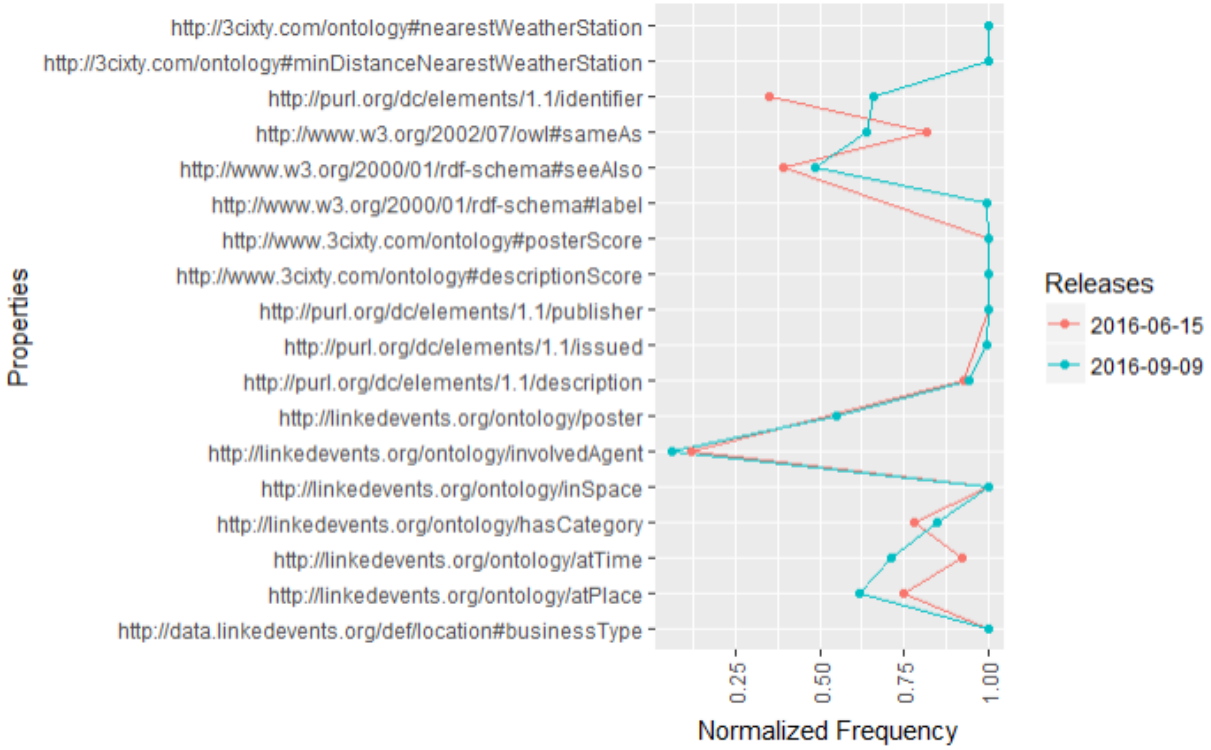


Figure 12. 3cixty *lode:Event* completeness measure results

completeness measure, for each class, the total number of properties, the complete properties, the incomplete properties, and the percentage of complete properties. For example, *foaf:Person* class has a total of 396 common properties over the two considered versions. We computed the completeness measures over those 396 properties and identified 131 properties with completeness measure value of 0 (incomplete). The remaining 265 properties can be considered as complete. The percentage of complete properties can be computed as $(\frac{265}{396}) * 100 = 66.92\%$.

Discussion In general, the completeness measure is based on a pairwise comparison of releases. In this experimental analysis, we compared the last two releases to identify missing data instances in the last release. Below we summarize our findings:

– 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 instance, the *lode:Event* class property

Table 8
Properties for the DBpedia classes and Consistency measures. Results are based on Version 201604 with threshold T=100.

Class	Total	Inconsistent	Consistent
dbo:Animal	162	123	39
dbo:Artist	429	329	100
dbo:Athlete	436	298	138
dbo:Film	450	298	152
dbo:MusicalWork	325	280	45
dbo:Organisation	1014	644	370
dbo:Place	1,090	589	501
dbo:Species	99	57	42
dbo:Work	935	659	276
foaf:Person	381	158	223

lode:atPlace[20] exhibits an observed frequency of 1632 in release 2016-06-15, while it is 424 in

[20] `http://linkedevents.org/ontology/atPlace`

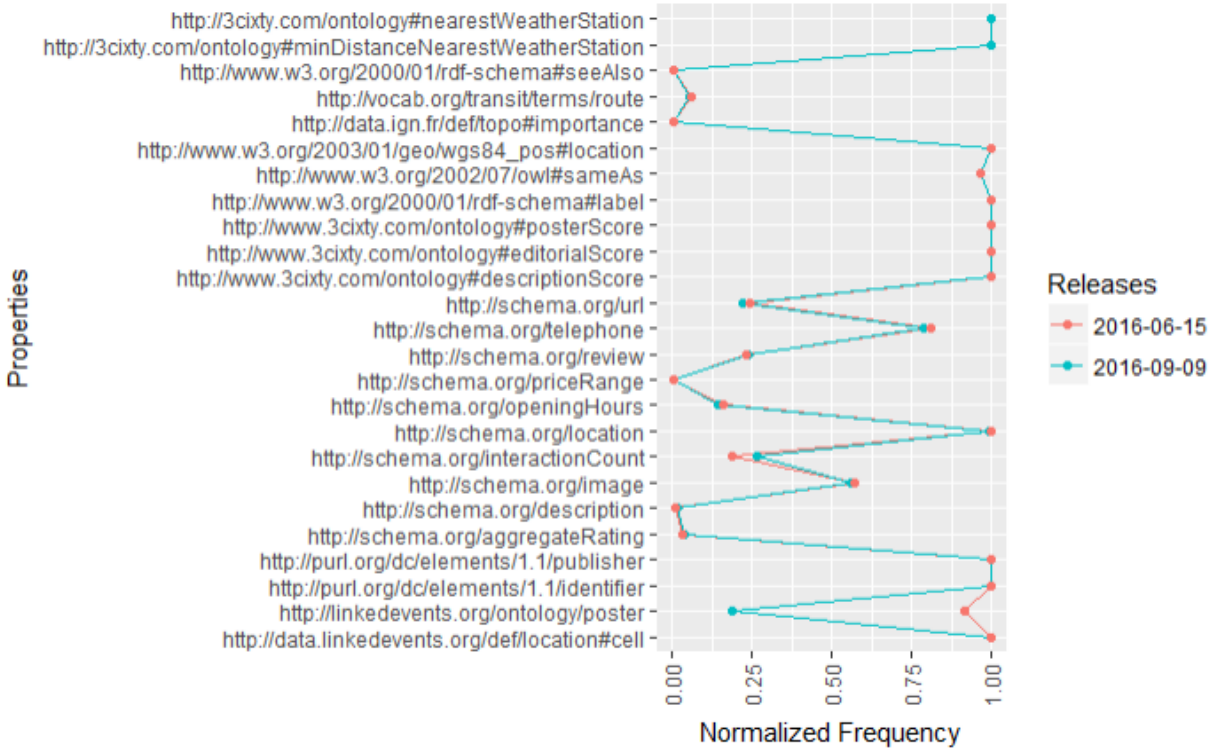


Figure 13. 3cixty *dul:Place* completeness measure results

Table 9

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

Class	Properties	Incomplete	Complete	Complete(%)
dbo:Animal	170	50	120	70.58%
dbo:Artist	372	21	351	94.35%
dbo:Athlete	404	64	340	84.16%
dbo:Film	461	34	427	92.62%
dbo:MusicalWork	335	46	289	86.17%
dbo:Organisation	975	134	841	86.26%
dbo:Place	1,060	141	920	86.69%
dbo:Species	101	27	74	73.27%
dbo:Work	896	89	807	90.06%
foaf:Person	396	131	265	66.92%

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*-type percentage of completeness is

50%, such a figure indicates a high number of incomplete predicates in the latest version (2016-09-09).

– 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 9. For instance, we identified a total of 131 incomplete properties for *foaf:Person* class. The *foaf:Person* class property *dbo:firstRace* 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 (foaf:Person) class percentage of completeness is 66.92%, such figure indicates a high number of incomplete instances in the last release (201604).

6.3. Qualitative Analysis

The general goal of our study is to verify how the *evolution 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.

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 representativeness of the experimental data. In particular we considered a random subset of entities. In general, a quality issue can identify a potential error in the KB. In this account, we focused on the effectiveness of the quality measures when it is able to detect an actual problem in the KB.

In general, the goal of this step is to extract, inspect, and perform manual validation for identifying the causes of quality issues. More in details, manual validation tasks are based on the following four steps:

i) Instances: we have selected a portion of the properties with quality issues from the quantitative analysis. The proposed quality characteristics are based on the results from statistical profiling. However, for manual validation we have extracted all the entities from both versions of a given KB. Then, we performed a set of disjoint operations to identify those missing entities in the last release of the KB.

ii) Inspections: using the dataset from instance extraction phase, we explored each missing instances for manual validation and report. In general, KBs use automatic approaches to gather data from the structured or unstructured data sources. For example, DBpedia KB uses an automatic extraction process based on the mapping with Wikipedia pages. For the manual validation, we have inspected the sources using the missing instances to identify the causes of quality issues. In particular, we manually checked if the information is present in the data sources but missing in the KB.

iii) Report: the validation result of a entity is reported as true positive (the subject presents an issue, and an actual problem was detected) or false positive (the item presents a possible issue, but none actual problem is found).

In particular, using the interpretation criteria reported in Table 4, 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 the precision for evaluating the effectiveness of our approach. Precision is defined as the proportion of accurate results of a quality measure 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:

$$p = \frac{TP}{TP + FP}. \tag{1}$$

We evaluated the precision manually by inspecting the results marked as issues from the completeness and consistency measures. For persistency and historical persistency, we have investigated a subset of resources for an entity type. The primary motivation is to detect the causes of quality issues for that entity type. Furthermore, historical persistency is a derived measure from persistency therefore we only performed the validation for persistency.

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:Place*, *dbo:Species*, *dbo:Film* 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 10, we present an overview of our selected classes and properties along with the experiments and, in Table 11, we summarize the manual evaluation results.

Table 10
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* and *dbo:Film* class to verify persistency and historical persistency
	Property	*foaf:Person* and *dbo:Place* class 50 properties from completeness measure to verify as well as compute precision.
	Property	*foaf:Person* class 158 properties and *dbo:Place* class a subset of 114 properties from consistency measure to verify as well as compute precision.

Persistency & Historical Persistency We evaluated the persistency measure based on the number of entity counts for *lode:Event*-type between two different 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* and *dbo:Film* 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 class *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 detail the first six missing entities. For example, the entity *AIDS_II* [21] 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.
- dbo:Film*:* We performed fine grain analysis based on subset of entity type *dbo:Film* for the latest two releases of DBpedia (201510 and 201604). The counts are $177,989$ and $146,449$ respectively. We performed a comparison between the two releases to identify the missing entities; we found $49,112$ entities that are no more present in the latest release. We investigate in more detail the first six missing entities. For example, the subject *dbpedia:$9.99*in 201510 was classified with type *dbo:Work* as well as *dbo:Film*. However, in 201604 it has been removed from both *dbo:Work* and *dbo:Film* was removed. We further explore the Wikipedia page [22] and film exists. It is clearly an error in the data extraction in 201604 release.

[21] http://dbpedia.org/page/AIDS_(computer_virus)
[22] https://en.wikipedia.org/wiki/\$9.99

Table 11
Summary of manual validation results

Characteristic	3cixty Nice	DBpedia	Causes of quality issues
Persistency & Historical Persistency	True positive: *lode:Event* entities missing due to algorithm error	False positive; *dbo:Species* and *dbo:Film* class quality issues fixed in current version;	*3cixty:* instances are missing due to an error in the reconciliation framework. *DBpedia:* erroneous schema presented in 201510 version has been fixed in 201604 version resulting in a false positive outcome.
Consistency	False Positive: *lode:Event* properties 2016-09-09 release we did not find any error	True positive ; *foaf:Person* and *dbo:Place* class on 201604 version we identify properties with consistency issue. Based on the threshold value of 100 it has a precision of 68% and 76%.	*3cixty:* In this use case, the schema remains consistent for all the KB releases and no real issues were found in the properties with low frequencies. *DBpedia:* In this use case, the schema evolves with each release. We found issues in the properties due to erroneous conceptualization.
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%. For *dbo:Place* over 50 properties we computed precision of 86%.	We found completeness issues due to data source extraction error for both *3cixty* KB and *DBpedia* KB.

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* and *dbo:Place* 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. In 3cixty KB schema remains consistent for all the releases. We identified that this properties common for all instances and we didn't find any erroneous conceptualization in the schema presentation.
- 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* 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 advocate for the creation of the *Quabbin Reservoir project*. However, the property relates to a person definition. This indicates an error present due to wrong mapping with Wikipedia Infobox keys. From the manual validation, the precision of the identified issues using the consistency measure accounts to 68%.
- dbo:Place *properties:* We have evaluated a total of 114 properties with consistency issues for *dbo:Place* class. We extracted all the data instances for the properties with consistency issues. From the manual inspection in *dbo:Place* class we identify data instances with erroneous conceptualization. For example, the property *dbo:weight* has 26 data instances mapped with *dbo:Place* type. We further investigate each of this data instances and corresponding Wikipedia pages. From manual investigation we can identity *dbo:weight* property erroneously mapped with *dbo:Place* type. Such as one of the data instance *wikipedia-en:Nokia_X5* is about mobile devices is mapped with *dbo:Place* type. This indicates an inconsistency issue due to wrong schema presentation. Based on the manual validation results we evaluate precision of 76% for *dbo:Place* class.

Completeness For the 3cixty KB, we analyzed the 2016-06-06 and 2016-09-09 releases; we evaluated the properties attached to *lode:Event* entities. DBpedia KB entity type of *foaf:Person* and *dbo:Place* in 201510 and 201604 releases has 131 and 437 properties with 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:* We have randomly selected 50 properties from *foaf:Person* class which is identified as incomplete in the quantitative experiment. In our manual inspection, we investigated a small number of the subjects presented in each property. More specifically, we first checked five subjects for manual evaluation for each property. 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 201604 has only 16 instances and for version 201510 has 217 instances. We performed a entities comparison between these two releases to identify the missing instances of the given property *dbo:bnfId* in the 201604 release. After a comparison between the two releases, we found 204 distinct instances missing in 201610 version of DBpedia. We perform a further manual investigation on the instances to verify the result.
  One of the results of the analysis is *John_Hartley _(academic)*[23] 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)*[24]. In the Wikipedia page *BNF ID* is present as linked to external source. In DBpedia from 201510 version to 201604 version update, this entity has been removed from the property *dbo:bnfId*. This example shows a completeness issue presents in the 201604 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%.
- dbo:Place *properties:* From the incomplete properties list of *dbo:Place* class we randomly selected 50 properties. We checked first five entities for manual evaluation. For *dbo:Place* class, we checked a total of 250 entities. For example, we identified that the property *dbo:parish* has completeness issue. We extracted all the instances for the releases of 201510 and 201610. Then we perform manual inspection for each entity and compared with the Wikipedia sources to identify the causes of quality issues.
  For example, property *dbo:parish* has 26 entities for 201510 and 20 entities in 201604. We collect missing resources after performing set disjoint operation. One of the results of the set disjoint operation is *Maughold_(parish)* missing in the 201604 version. To further validate such an output, we checked the source Wikipedia page using *foaf:primaryTopic* about *wikipedia-en:Maughold_(parish)*. In the Wikipedia page *Parish* is presented as title definition of the captain of parish militia. In particular, in DBpedia from 201510 version to 201604 version update, this entity has been removed from the property *dbo:parish*. Based on the investigation of the properties, we compute our completeness measure has the precision of 86%.

7. Discussion

In this paper, we have proposed an approach to detect quality issues leveraging four quality characteristics derived by KB evolution analysis. In this section, we first summarize our findings. Then, we discussed the limitations that we have identified and outlined the planning of future activities.

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

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

7.1. Evolution Analysis to Drive Quality Assessment

Similarly to Radulovic et al. [37], we present a discussion of our approach with respect to the following four criteria:

Conformance provides insights to what extent a quality framework and characteristics meet established standards. In our approach, we have proposed four quality characteristics. Among them we selected the completeness and consistency quality characteristics according to the guidelines from the ISO 25012 standard. On the other hand, we followed the study presented by Ellefi *et al.* [7] to propose persistency and historical persistency quality characteristics.

Applicability implies the practical aspects of the quality assessment approach. In general, coarse-grained analysis significantly improves the space and time complexity regarding data analysis. We envision that our approach can be automated using daily snapshot generations and automatically creating periodic reports. 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.

Causes of Quality Issues provides insights regarding detected issues using our approach. In our approach, we identified two types of quality issues: *i)* errors in the data source extraction process, and *ii)* erroneous schema presentation. In the case of 3cixty Nice KB, we only found issues based on the data source extraction process. For example, we found a significant number of resources missing in the last release of *lode:Event* class due to algorithmic error. On the other hand, 3cixty Nice KB schema remains unchanged in all the KB releases. More specifically, we didn't find any real issues based on the schema presentation in regarding consistency measure. In the case of DBpedia KB, we found both types of quality issues. For example, entities missing in *foaf:Person* class due to incorrect mapping of field values in the data extraction process. For example, we found a significant number of resources missing due to wrong schema presentation for the DBpedia KB. Such as property *dbo:Lake* mapped with *foaf:Person*-type due to automatic mapping with wrong Wikipedia infobox keys. Based on the two use cases, our approach has proven highly efficient to identify quality issues in the data extraction and integration process.

Performance We evaluated our quality assessment approach in terms of precision through manual evaluation. We present persistency measure only to monitor the stability of a class instead of fine-grain analysis on the entities. In particular, the quality characteristic which is measured at the class level such as persistency, we only investigated the detected quality issue is true positive (TP) or false positive (FP). Based on the qualitative analysis, persistency measure for the 3cixty Nice KB has TP results, and the DBpedia KB has FP results. On the other hand, we have evaluated precision based on completeness and consistency measures for the both KBs. The computed precision of completeness measure in our quality assessment approach is: *i)* 94% for *foaf:Person*-type entities of DBpedia KB; *ii)* 86% for *dbo:Place*-type entities of DBpedia KB, and *iii)* 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 and 76% for *dbo:Place*-type entities.

7.2. 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 DBpedia Live [25] and 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

[25] http://wiki.dbpedia.org/online-access/DBpediaLive

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.

7.3. 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 *dbo:bnfId* 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 *dbo:bnfId* in details on the 201604 release. We explored the property description that leads to Wikidata link[26] 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 12, 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)*[27] 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.

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

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

Table 12
A sample of 6 subjects and objects of *bnfId* property

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"

7.4. Lifespan analysis of Evolving KBs

On the basis of the dynamic feature [7], 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 the DBpedia KB, we observed that variations in the knowledge base growth could affect quality issues. Furthermore, we argue that quality issues can be identified through monitoring lifespan of an RDF KBs.

We can measure growth level of KB resources (instances) by measuring changes presented in 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 lifespan analysis regarding change patterns over time as well as experiments on the 3cixty Nice KB and the DBpedia KB. To measure the KB growth, we applied linear regression analysis of entity counts over KB releases. In the regression analysis, we checked 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 $i = 1...n$ present as the 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 the KB growth of a class C as:

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

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

3cixty Nice case study The experimental data is reported in Table 5. 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 14a and 14b.

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 $i = 1...n$.

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, for the *dul:Place* class, $ND_{places} < 1$ so the KB growth measure value =0 .

In the case of 3cixty Nice KB, the *lode:Event* class clearly presents anomalies as the number of distinct entities drops significantly on the last release. In Figure 14a, the *lode:Event* class growth remains constant until it has errors in the last release. It has higher distance between actual and predicted value based on the *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.

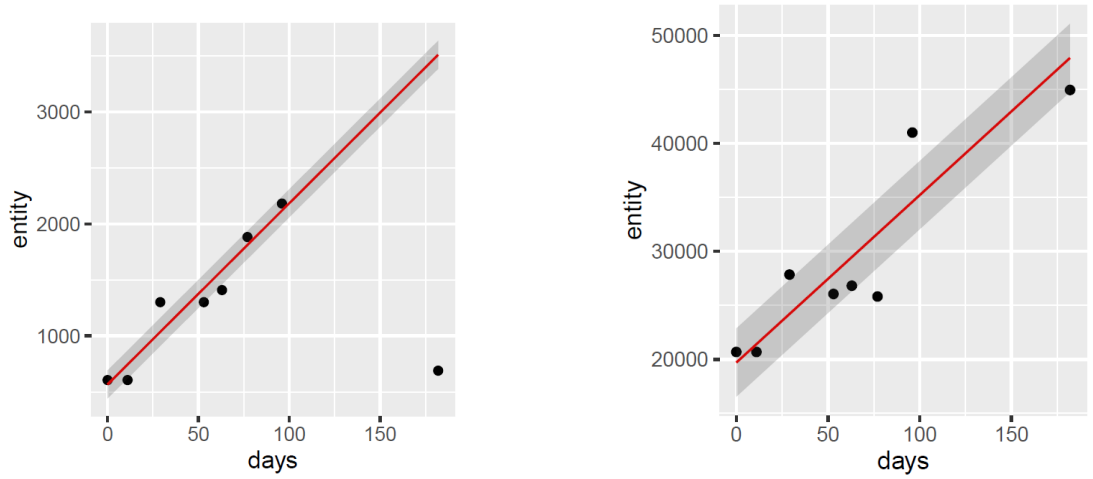
DBpedia Case study The experimental data is reported in Table 6. Based on the KB growth measure definition, we measured the normalized distance for each class (Table 13). We compared with the number of entities from the last release (201604) actual value and predicted value from the linear regression to measure the normalized distance. From the results observed for *dbo:Artist*, *dbo:Film*, and *dbo:MusicalWork*, the normalized distance is near the regression line with $ND < 1$. In Figure 15, 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.

Table 13
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

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 high-level changes by analyzing KB growth patterns. Data curators can identify persistency issues in KB resources using lifespan analysis. However, a further exploration of the KB lifespan analysis is needed, and we



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

Figure 14. 3cixty two classes KB growth measure

consider this as a future research activity. In particular, we want to explore further *(i)* which factors are affecting KB growth and *(ii)* validating the stability measure.

7.5. Limitations

We have identified the following two limitations.

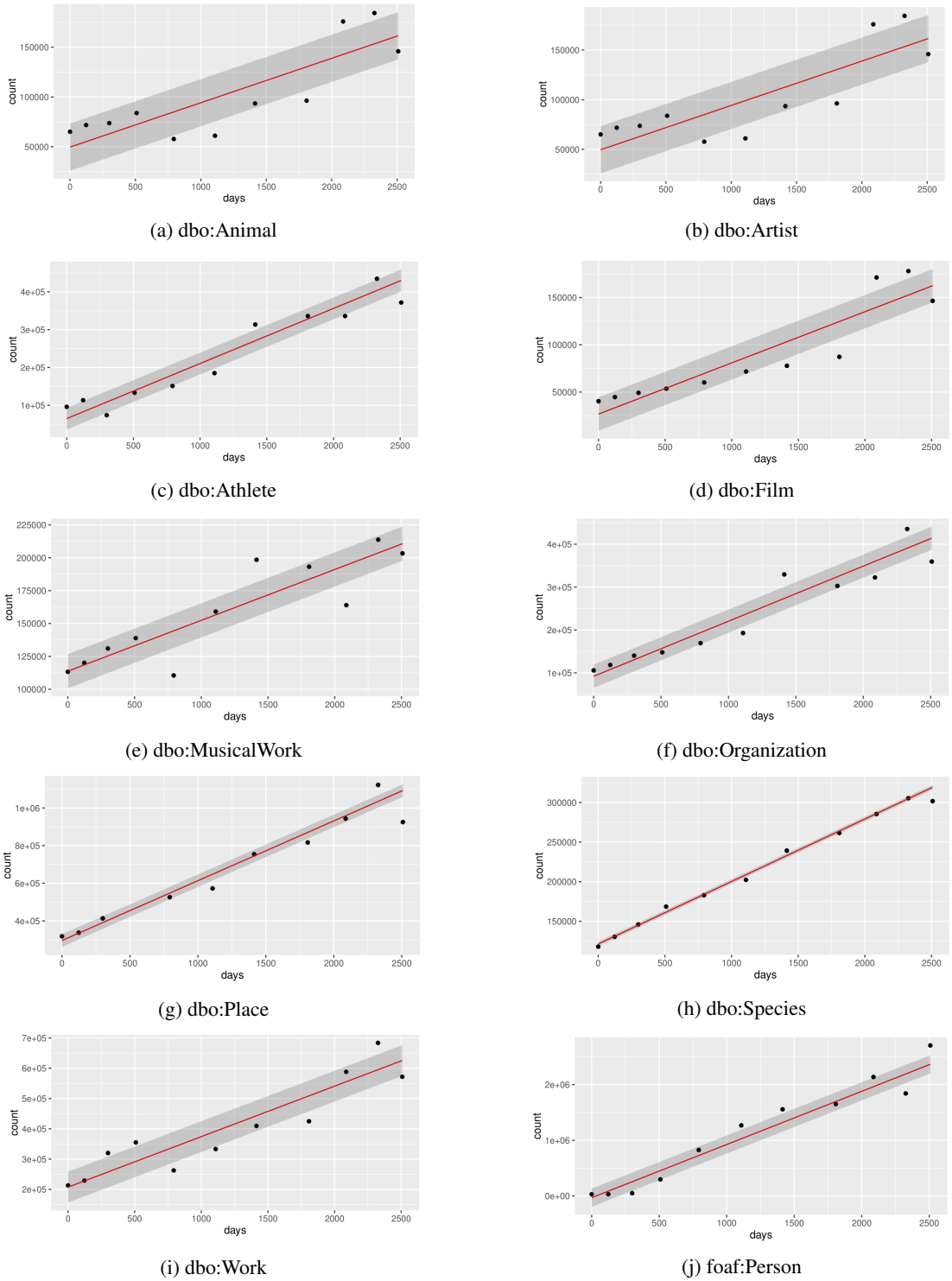
First, as a basic measurement element, we only considered aggregated measures from statistical profiling such as frequency of properties in a class. For the qualitative analysis, we considered raw knowledge base differences among releases. In order to detect actual differences, we would need to store two releases of a KB in a single graph, and perform the set difference operation. We performed manual validation by inspecting data sources. However, this approach of qualitative analysis has several drawbacks from a technical point of view. Furthermore, regardless of the technical details, the set difference operation is, computationally-wise, extremely expensive. As a future work, we plan to extend our manual validation approach by cross-referencing GitHub issues or mailing lists.

Second, KBs are growing over time with new resources that are added or deleted. In this study, we only considered the negative impact of erroneous deletion of resources. As a future work, we plan to investigate the negative impact of the erroneous addition of resources in the KBs.

8. Conclusion and Future Work

The main motivation for the work presented in this paper is rooted in the concepts of Linked data dynamics[28] 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 [20]. However, less focus has been given toward understanding knowledge base resource changes over time to detect anomalies over various releases. To verify our assumption, we proposed four quality characteristics, based on the evolution analysis. We designed a quality assessment approach by profiling quality issues using different Knowledge Base (KB) releases. In particular, we explored the benefits of aggregated measures using quality profiling. The advantage of our approach lies in the fact that it captures anomalies for an evolving KB that can trigger alerts to the data curators in the quality repairing processes. More specifically, we consider coarse-grained analysis as an essential requirement to capture any quality issues for an evolving KB. Although coarse-grained analysis cannot detect all possible quality issues, it helps to identify common quality issues such as erroneous deletion of resources in the data extraction and integration processes. Moreover, the drawback of fine-grained analysis using raw change detection is the significant space and time complexity.

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



(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

Figure 15. DBpedia 10 classes KB growth measure

In this paper, we proposed completeness and consistency quality characteristics from the ISO 25012 standard. Also, we proposed persistency and historical

persistency quality characteristics based on dynamic features presented by Ellefi *et al.* [7]. 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 issues 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 the 3cixty Nice KB, remain stable in case of the consistency issue. On the other hand, KBs with low-frequency updates (monthly or yearly updates), such as DBpedia KB, seem to have more inconsistency issues.
- 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 are as follows:

- We plan to expand our evolution based quality analysis approach by analyzing other quality characteristics presented in literature such as Zaveri *et al.* [47]. Also, we intend to apply our approach to KBs in other domain to further verify our assumption;
- A limitation of the current approach is that we only considered negative impact of deletion of resources. We plan to study how we can dynamically adapt impact of addition of resources in a KB;
- We chose 100 since from the empirical analysis at property level it allowed to maximize the precision of the approach. However, the process of threshold value selection needs further investigation. As a future work, we plan to perform additional statistical analysis to motivate the choice of this threshold.
- 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 lifespan of a KB. This has led to conceptualize the Stability quality characteristics, which is meant to detect anomalies in a KB. We plan to monitor various KB growth rates 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. Luzzu—a methodology and framework for linked data quality assessment. *Journal of Data and Information Quality (JDIQ)*, 8(1):4, 2016.

[6] Renata Dividino, Ansgar Scherp, Gerd Gröner, and Thomas Grotton. Change-a-lod: does the schema on the linked data cloud change or not? In *Proceedings of the Fourth International Conference on Consuming Linked Data-Volume 1034*, pages 87–98. CEUR-WS. org, 2013.

[7] Mohamed Ben Ellefi, Zohra Bellahsene, J Breslin, Elena Demidova, Stefan Dietze, Julian Szymanski, and Konstantin Todorov. Rdf dataset profiling-a survey of features, methods, vocabularies and applications. *Semantic Web*, 2017.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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/.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Joakim Lindblad. Histogram thresholding using kernel density estimates. In *In Proceedings of the Swedish Society for Automated Image Analysis (SSAB) Symposium on Image Analysis, Halmstad, Sweden*, pages 41–44, 2000.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Felix Naumann. Data profiling revisited. *SIGMOD Rec.*, 42(4):40–49, February 2014.

[31] Chifumi Nishioka and Ansgar Scherp. Information-theoretic analysis of entity dynamics on the linked open data cloud. In *PROFILES@ ESWC*, 2016.

[32] Jack E. Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2003.

[33] 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.

[34] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[35] Heiko Paulheim and Christian Bizer. Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):63–86, 2014.

[36] Nathalie Pernelle, Fatiha Saïs, Daniel Mercier, and Sujeeban Thuraisamy. Rdf data evolution: efficient detection and semantic representation of changes. In *Semantic Systems-SEMANTiCS2016*, pages 4–pages, 2016.

[37] 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.

[38] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

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

[45] Max Völkel and Tudor Groza. Semversion: An rdf-based ontology versioning system. In *Proceedings of the IADIS international conference WWW/Internet*, volume 2006, page 44, 2006.

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

[47] 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.