KBQ - A Tool for Knowledge Base Quality Assessment Using Evolution Analysis

Mohammad Rashid
Politecnico di Torino, Italy
mohammad.rashid@polito.it

Giuseppe Rizzo
Istituto Superiore Mario Boella, Italy
giuseppe.rizzo@ismb.it

Nandana Mihindukulasooriya
Universidad Politécnica de Madrid, Spain
nmihindu@fi.upm.es

Marco Torchiano
Politecnico di Torino, Italy
marco.torchiano@polito.it

Oscar Corcho
Universidad Politécnica de Madrid, Spain
ocorcho@fi.upm.es

ABSTRACT

Knowledge bases are becoming essential components for tasks that require automation with some degrees of intelligence. It is crucial to establish automatic and timely checks to ensure high-level quality of the knowledge base content (i.e., entities, types, and relations). In this paper, we present KBQ, a tool that automates the detection and report generation of quality issues for evolving knowledge bases. KBQ analyzes the evolution of a KB by measuring the frequency of change, the change pattern, the change impact and the causes of changes of resources and properties. Data collection and profiling tasks are performed using Loupe, an online tool for linked data profiling. We describe KBQ in action on two different use cases, and we report the benefits that it introduced. KBQ is published as open source project, and a demo is available at http://datascience.ismb.it/shiny/KBQ/.

CCS CONCEPTS

• **Information systems → Data cleaning**; • **Computing methodologies → Knowledge representation and reasoning**;

KEYWORDS

Knowledge Base, Linked Data, Quality Assessment, Quality Issues, Evolution Analysis

1 INTRODUCTION

In the recent year's much efforts have been given towards sharing Knowledge Bases (KB) in the Linked Open Data (LOD) cloud[1]. Popular knowledge bases such as DBpedia, YAGO2, and Wikidata have chosen the RDF data model[2] to represent their data due to its capabilities for semantically rich knowledge representation. RDF KBs are evolving since both data instances, and schemes are updated, extended, revised and refactored covering more and more topical domains [8]. In particular, entities evolve given that new data is added, old is removed, and links to entities are updated or deleted. Within this context, *data quality* for evolving KBs remains a critical aspect to obtain trust by the users. Data quality, in general, relates to the perception of the "fitness for use" in a given context [22]. Manual quality assessment and representation of large KBs is neither feasible nor sustainable [6]. On the other hand, assessing continuously and automatically the quality of a knowledge base is a challenging task as data is derived from many autonomous, evolving, and increasingly large data providers.

Various tools have been developed for linked data quality assessment based on manual, semi-automatic, and automated approaches. For example, TripleCheckMate[3] is a crowdsourced quality assessment tool focusing on the correctness of the DBpedia resources. RDFUnit [9] is a tool centered around the definition of data quality integrity constraints. Flemming's [4] data quality assessment tool calculates data quality scores based on manual user input for data sources. Debattista *et al.* describe a conceptual methodology for assessing Linked Datasets, proposing Luzzu [1], a framework for Linked Data Quality Assessment. Although these tools guarantee an appropriate data quality assessment, less focus has given towards evolution aspects of a KB. In particular, these tools did not considered the impact of KB evolution such as capture the changes that indicates an abnormal situation or changes that the curator wants to highlight because they are useful for a specific domain [15].

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 [16]. Data profiling is defined as the process of examining data to collect statistics and provide relevant metadata about the data [14]. Based on data profiling we can thoroughly examine and understand each KB, its structure, and its properties before usage. *Evolution analysis* using dynamic feature help to understand the changes applied to an entire KB or parts of it. In general, the dynamic feature of a dataset gives insights into how it behaves and evolves over a certain period [15]. Ellefi *et al.* [2] explored the dynamic features for data profiling considering the use cases presented by Käfer *et al.* [8]. They present dynamic features in multiple dimension regarding the KB update behavior, such as frequency of change, changes pattern, changes impact and causes of change.

In this paper, we present KBQ a tool for KB quality assessment using evolution analysis. One of the core ideas in this work is to use dynamic features from data profiling results for analyzing the KB evolution. Our quality assessment approach based on two main areas: (1) evolution of resources and (2) impact of the unwanted removal of resources in a KB. In particular, based on the detected

[1]http://lod-cloud.net
[2]https://www.w3.org/RDF/
[3]http://aksw.org/Projects/TripleCheckMate.html

changes between various releases, we aim to analyze and validate quality issues in the KBs.

ISO/IEC 25012 [7] standard defines data quality as the degree to which a set of characteristics of data fulfills requirements. Data quality issues are the specific problem instances that we can find issues based on quality characteristics and prevent data from being regarded as high-quality [10]. More specifically, quality characteristics are abstract definition indicating quality issues. In this work we explored two main quality issues, namely *lack of persistency* and *lack of completeness*:

Lack of Persistency relates to resources that were present in a previous KB release, but then they disappeared. In particular, look into the problem due to unexpected removal of information.

Lack of Completeness refers to the problem due to incomplete resources present in a knowledge base; this happens due to systematic errors in data extraction and integration processes.

In KBQ, quality assessment is performed by four quality characteristics, such as persistency, historical persistency, completeness and KB growth. We use basic statistics (i.e., counts, and diffs) of entities, types, and relations over the extracted triples from various releases for measurement function.

KBQ builds upon the data collection and profiling functionalities of Loupe [13], an online system that inspects and extracts automatically statistics about the entities, vocabularies used (classes, and properties), and frequent triple patterns of a KB. We created a set of APIs[4] for periodic snapshots generation and maintaining scheduled tasks for automatic and timely quality assessment. In this paper, we describe KBQ in action with `lode:Event`[5] entity in the 3cixty [23] KB and `dbo:Place`[6] entity in the DBpedia [11] KB, reporting the benefits introduced to the corresponding projects.

2 EVOLUTION-BASED QUALITY CHARACTERISTICS

Data quality is a cross-disciplinary and multidimensional concept. According to Pipino *et al.* [20], based on context, quality can be both subjective perceptions and objective measurements. Quality measurement function are based on dynamic features from data profiling results. The quality indicators are weighted values, which give the freedom to define multiple degrees of importance [4]. In our approach, the quality indicators are based on the changes present at the statistical level in terms of variation of absolute and relative frequency count of entities and predicates between pairs of KB release. We formalized each quality indicator values in the range $[0, 1]$. We considered four quality characteristics for quality assessment tasks, namely *Persistency*, *Historical Persistency*, *Completeness* and *KB growth*.

2.1 Persistency

Knowledge Bases contain information about different real-world objects or concepts commonly referred as entities. In general, quality issues regarding unexpected removal of information from current version may impact the stability of the KB. Persistency characteristics help to understand stability feature. Ellefi *et al.* [2] present stability feature as an aggregation measure of the dataset dynamics. It helps to understand to what extent the performed update impacts the overall state of the knowledge base. In particular, it provides insights into whether there are any missing resources in the last KB release.

Quality Indicator: It is a class specific measure and measurement function based on the entity count difference between two KB releases. We compute the persistency measure value of 0 if the entity count of the last version is lower than the previous version otherwise 1. The value of 1 implies no persistency issue present in the class. The value of 0 implies persistency issues found in the class.

2.2 Historical Persistency

Historical persistency is a derived measure based on persistency characteristics. It measures the lifespan of an entity type. Ellefi *et al.* [2] present lifespan feature based on the degree of changes. The degree of changes capture the impact of changes observed on an entire dataset or parts of it. Also, lifespan represents the period when a certain entity is available. In particular, this value gives an overview of persistency issues present in an entity type over all releases. It helps data curators to decide which knowledge base release can be used for future data management tasks.

Quality Indicator: The Historical Persistency measure evaluates the persistency over the history of the KB and is computed as the average of the persistency measures for all releases. High percentage implies an estimation of fewer issues and lower percentage entails more issues present in KB releases.

2.3 Completeness

This measure focuses on the removal of information as a negative effect of the KB evolution. Zaveri et al. [25] refer to completeness as the degree to which all required information is present in a particular dataset. They present completeness characteristics based on the following four 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. We considered the aspects of property completeness for KB evolution.

Quality Indicator: The basic measure we use is the difference between the frequency of properties for a class between two KB releases. In particular, if the instance count of properties present in the class has negative count compare to the previous release then we assume there is a completeness issue. We assign value of 1 if no completeness issues are present while value of 0 entails none completeness issue is present. Also at the class level, we compute the percentage of completeness based on the number of completeness issue divided by total properties.

[4]The src code is available at https://github.com/rifat963/KBDataObservatory
[5]http://linkedevents.org/ontology/Event
[6]http://dbpedia.org/ontology/Place

2.4 KB growth

In this measure, we explore the aspect of KB growth by measuring the growth level of KB resources (instances) over the different releases. Ellefi *et al.* [2] present growth rate feature as the level of growth of a dataset in terms of data instances. In particular, KB growth explores the change patterns of a knowledge base. Change patterns help to understand the existence and kinds of categories of updates or change behavior. It can help to understand changes present in the KB has upward or downward trend. We assume that if the schema remain consistent then downward trend at the last release may indicate a potential problem in the data extraction process.

Quality Indicator: We use a simple linear regression model to predict the KB growth level of resources. It is a class specific measure and measurement function based on the entity count from all the KB releases. Using the difference between the observed and predicted entity count values at tbe last KB release, we can detect the trend in the KB growth level. We evaluated the normalized distance based on the entity type residual value divided by mean residual value. We used normalize distance between observed and predicated entity count value to measure KB growth. In particular, if the normalized distance is greater than 1 then the KB may have unexpected growth with unwanted entities otherwise KB remains stable.

Table 1: Quality Indicators

Quality Characteristics	Quality Indicators	Interpretation
Persistency	Persistency measure values [0,1]	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 [0,1]	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.
KB growth	KB growth measure value [0,1]	The value of 1 implies no unexpected growth present in the class. The value of 0 indicated that unexpected growth may happen in the current version of the class.

3 ARCHITECTURE OVERVIEW

KBQ is composed of four modules that are illustrated in Fig. 1. We implemented KBQ using the R statistical package that we share as open source in order to foster reproducibility of the experiments[7]. The modules are explained in detail below.

Collect: generates knowledge base (KB) snapshots and sets up timely schedulers. It supports (i) collection of KB summary statistics via a dedicated SPARQL endpoint; this component is built on top of Loupe [13]; (ii) collection of periodic KB snapshots that are accessible through a SPARQL endpoint saved in a CSV files. We named each CSV file based on the entity type. In particular, we used SPARQL endpoint as an input and save the results extracted from the SPARQL endpoints into CSV files.

Analyze: performs quality profiling based on a particular entity type and generates quality problem report. We build an intermediary data structure by grouping sets of resources and predicates for a entity type based on KB releases to speed up the execution of the measurement functions. We use the values of quality measures as indicators for the quality issues. In Table 1, we present the quality indicators used in our tool. This module allows saving the analyses to an HTML file.

Visualize: is composed of two modules: (i) list of quality assessment results and (ii) data set catalogue. Visualization of quality assessment results are embedded with analysis module based on four quality characteristics. This allows any user to access quality measures by selecting a specific characteristics. It also allows class faceted exploration along the various KB releases.

Validate: extracts, inspects and allows manual annotations of quality issues. A user can extract properties with quality issues after performing a quality profiling that consists of: *(i)* Incomplete properties: visualize a list of properties with completeness quality issues for validation. *(ii)* Instances: quality profiling is done based on summary statistics. To extract the missing instances of a property, the instance extraction component performs comparison between the list of instances from the last two versions. *(iii)* Inspections: after the instance extraction is done, a user can select every instance for inspection and report. We present instance inspection based on data sources. In particular, validation is performed by inspecting the missing instances and manually evaluate cause of quality issues through data source inspections. *(iv)* Report: a user can report if the instance is 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), as well as a user can comment on specific issues. Finally, a user can save the validation report in a HTML file.

[7] https://github.com/KBQ/KBQ

4 USE CASES: KBQ IN ACTION

We present KBQ in action for 3cixty KB [23] and Spanish DBpedia KB [11]. We selected these two KBs according to: *i)* popularity and representativeness in their domain: DBpedia for the encyclopedic domain, 3cixty 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. A recorded video of KBQ in action for these two use cases is available at https://youtu.be/F02l7ImOZV8.

4.1 3cixty KB quality assessment

3cixty KB is continuously changing with frequent updates (daily updates). We target `lode:Event` class for quality profiling. Using KBQ we manually collected 9 snapshots from 2016-03-11 to 2016-09-09. In addition, we collected daily snapshots starting from 2017-07-19 till 2017-09-27 using the scheduler. Overall, this results: (1) For both manually saved snapshots and scheduler generated ones, the

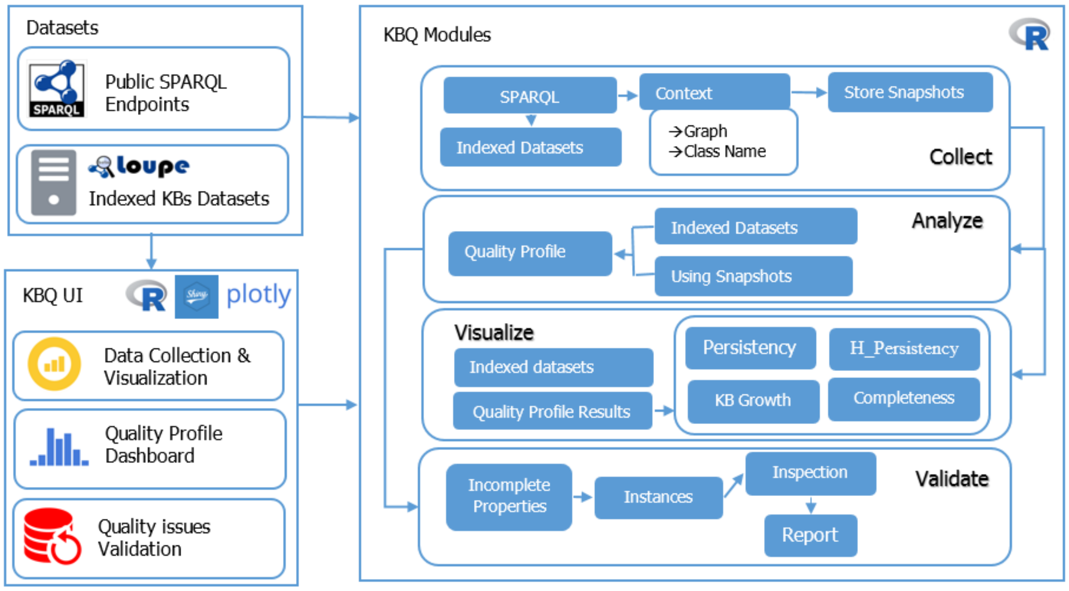

Figure 1: High level architecture of the KBQ tool.

Persistency measure value of 1 indicates no missing entities in the last version of `lode:Event` class. (2) Historical persistency value of 87.5% for manually saved snapshots estimates little variation presents over all releases. Persistency issues are present only between release 2016-06-16 and 2016-09-09. Also using the scheduler measure value of 85.7% estimates small variation presents where persistency issue is only present between 2017-07-22 and 2017-07-23. (3) Completeness measures for manually saved snapshots on last releases of 2017-07-19 detected two properties with quality issues. (4) KB growth monitors the dynamics of knowledge base changes. For manually saved snapshots of `lode:Event`, the value of 0 indicates higher growth than expected on the last release. Furthermore we validated the quality using validation module.

In Figure 2 we present the persistency quality assessment results for lode:Event class. Finally, we save quality profiling results in a HTML file (example of a generated report[8]).

4.2 Spanish DBpedia quality assessment

Spanish DBpedia has less frequent updates (monthly or yearly updates). We target `dbo:place` class for quality profiling. We collected summary statistics of 11 different releases for Spanish DBpedia. The quality profiling results of `dbo:place` class: (1) Persistency value of 1 indicates no missing entities in the last version. (2) Historical persistency has 100% indicating consistent growth across releases. (3) Completeness: in version 201610 of DBpedia we detected 9 properties with quality issues. (4) KB growth of `dbo:place` is equal to 0 indicating higher growth (over the expected) on the last release. In Figure 3 we present the persistency quality assessment results for dbo:place class. Finally, we save quality profiling results in a HTML file (example of a generated report[9]).

[8]http://datascience.ismb.it/shiny/2017-07-21-QualityProblemReport.html
[9]http://datascience.ismb.it/shiny/2017-07-21--http---dbpedia.org-ontology-Place-.html

4.3 Discussion

We identified a set of properties with quality issues using evolution based quality characteristics from the Spanish DBpedia KB and 3cixty KB. Furthermore, we evaluate the results from quality analysis using manual validation approach. We performed the manual validation based on the detected missing properties from `dbo:Place` class. For a selected property validation module collect all instances presents in the last two releases. For example, we selected the property `dbo:prefijoTelefóicoNombre` [10] to be manually validated. We used KBQ to collect all the instances (56109,55387) from the two releases (201604,201610). Validation module performed a set disjointed operations between two triple sets to identify those triples missing from the 201610 release. From the set disjoint operation we found total 1982 instances missing from 201610 version. In order to inspect the missing instances we randomly select a subset of 200 instances for evaluation. From the manual evaluation, we identify `dbr:Morante`[11], which is available in the 201604 release. However, it is not found in 201610 release of DBpedia. In general, these instances are auto-generated from Wikipedia Infobox keys. To further validate them we track the Wikipedia page from which statement was extracted in the DBpedia KB. We checked the source Wikipedia page using *foaf:primaryTopic* about *Morante*[12]. In the Wikipedia page *prefijo TelefónicoNombre* is present in the Wikipedia infobox Key. In the Spanish DBpedia from 201604 version to 201610 version update, this data instance has been removed from the property *dbo:prefijoTelefóicoNombre*. Therefore, these instances are present in the Wikipedia Infobox as Keys but missing in the DBpedia 201610 release. This example shows a validity of the completeness issue presents in the 201610 release of DBpedia for property `dbo:prefijoTelefóicoNombre`.

[10]http://es.dbpedia.org/property/prefijoTelefóicoNombre
[11]http://es.dbpedia.org/page/Morante
[12]https://es.wikipedia.org/wiki/Morante

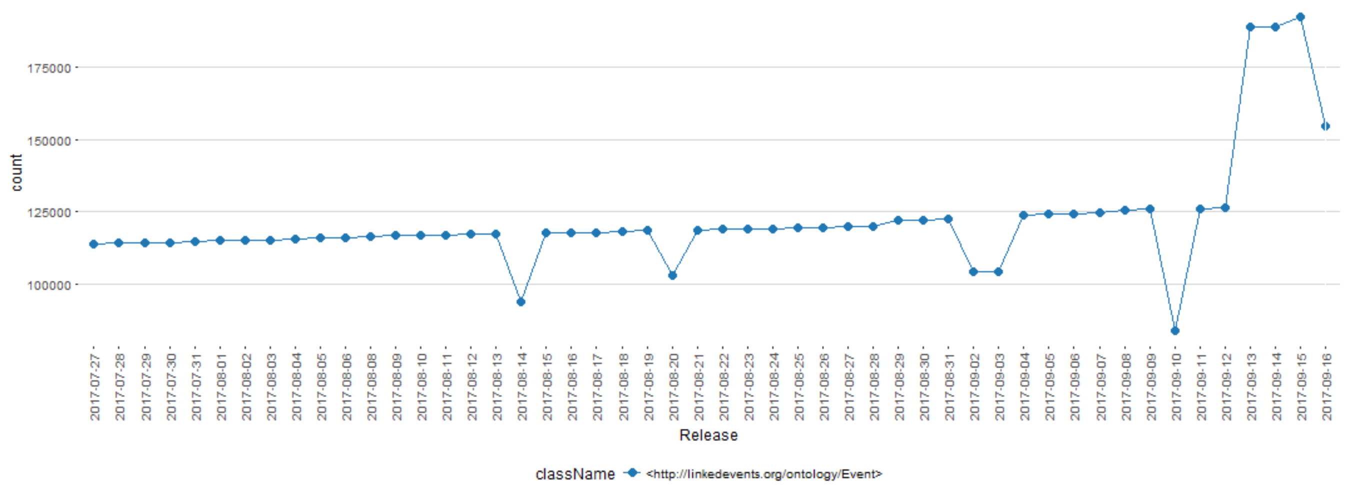

Figure 2: 3cixty lode:Event class.

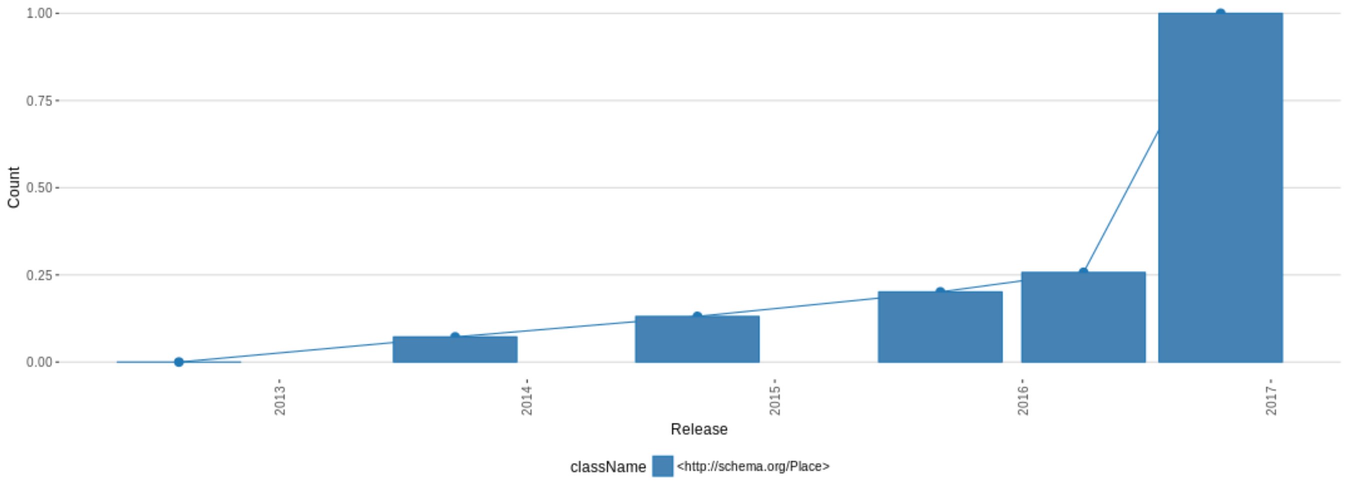

Figure 3: DBpedia dbo:place class.

5 RELATED WORK

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

Change Detection in Linked Data: There are various features of dataset dynamics which must be considered to achieve a comprehensive overview of how linked data changes evolve on the Web [8]. Issues in curated RDF(S) have been addressed by Papavasileiou et al. [17]. They introduce a high-level language of changes and its formal detection and application semantics, as well as a corresponding change detection algorithm, which satisfies these needs for RDF(S) KBs. Ellefi *et al.* [2] 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. Recently, Yannis et al. [21] proposed a framework that detected changes between versions. It enables easy and efficient navigation among versions, automated processing, and analysis of changes. They also include cross-snapshot queries (spanning across different versions), as well as queries involving both changes in schema and instance. Zabilith et al. [24] ontology conducted an extensive work at the ontology level detection, representation, and management of the changes. Pernelle et al.[19] present an approach which allows to detect and represent elementary and complex changes that can be detected only on the data level.

Linked Data Quality Assessment: Regarding the automated LOD quality assessment, Fleischhacker *et al.* [3] proposed a two-fold approach that relies on unsupervised outlier detection to identify numerical errors in objects of RDF triples. A probabilistic framework presented by Li *et al.* [12] that predicts arithmetic relations (equal, greater than, less than) among multiple RDF predicates to detect inconsistencies in numerical and date values. Based on the statistical distribution of predicates and objects in RDF datasets Paulheim *et al.*[18] presented two algorithms SDType and SDValidate. SDType predicts classes of RDF resource thus completing missing values of rdf:type properties. SDValidate detects incorrect links between resources within a dataset. The framework SWIQA proposed by Furber and Hepp [5] can be applied for detecting accuracy quality issues including incorrect object values, datatypes, and literals. These solutions are tailored to detect very specific errors in RDF triples. 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.

6 LIMITATIONS

We have identified the following two limitations, such as:

First, in this tool we detect changes between two KB releases only based on summary statistics. In particular, we applied coarse-grained analysis to capture any quality issues for evolving KB. Although coarse-grained analysis cannot capture all possible quality issues, it helps to identify common quality issues such as systematic errors in data extraction and integration processes.

Second, in KBQ we introduce a manual validation module where we aim to keep track of the detected quality issues by using true or false annotations. We aim to use this manually annotated result as a gold standard for future quality assessment tasks. Furthermore, we envision that an automatic schema validation using integrity constraints could be helpful for the validation process.

7 CONCLUSIONS AND FUTURE WORK

The main motivations for the work presented in this paper is rooted in the concepts of Linked data dynamics[13] on the one side and knowledge base quality on the other side. The focus of this work is to automate the timely process of quality issue detection without user intervention based on evolution analyses. More specifically, we explored the idea of monitoring KB changes as the premise of this work. We design and develop KBQ tool for Knowledge Base quality assessment using evolution analysis. We present four quality evolution based quality characteristics persistency, historical persistency, completeness and KB growth. KBQ is also knowledge base agnostic and we demonstrated its usage for two different use cases, namely 3cixty and Spanish DBpedia. In particular, in this work we explored the benefits of aggregated measures using quality profiling.

As future work, we plan to add automatic error annotations of the properties with quality issues. We also plan to extend our validation approach for automatic snapshots generation and publishing in a triple format.

ACKNOWLEDGMENTS

This work was partially funded by the Spanish government with the BES-2014-068449 grant and Datos 4.0 project.

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

REFERENCES

[1] Jeremy Debattista, SÓren Auer, and Christoph Lange. 2016. LuzzuA Methodology and Framework for Linked Data Quality Assessment. *Journal of Data and Information Quality (JDIQ)* 8, 1 (2016), 4.

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

[3] Daniel Fleischhacker, Heiko Paulheim, Volha Bryl, Johanna Völker, and Christian Bizer. 2014. Detecting errors in numerical linked data using cross-checked outlier detection. In *International Semantic Web Conference*. Springer, 357–372.

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

[5] Christian Fürber and Martin Hepp. 2011. Swiqa-a semantic web information quality assessment framework.. In *ECIS*, Vol. 15. 19.

[6] Christophe Guéret, Paul Groth, Claus Stadler, and Jens Lehmann. 2012. Assessing linked data mappings using network measures. *The Semantic Web: Research and Applications* (2012), 87–102.

[7] ISO/IEC. 2008. *25012:2008 – Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model*. Technical Report. ISO/IEC. http://iso25000.com/index.php/en/iso-25000-standards/iso-25012

[8] Tobias Káfer, Ahmed Abdelrahman, Júrgen Umbrich, Patrick OâĂŹByrne, and Aidan Hogan. 2013. Observing linked data dynamics. In *Extended Semantic Web Conference*. Springer, 213–227.

[9] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd international conference on World Wide Web*. ACM, 747–758.

[10] Nuno Laranjeiro, Seyma Nur Soydemir, and Jorge Bernardino. 2015. A survey on data quality: classifying poor data. In *Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on*. IEEE, 179–188.

[11] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[12] Huiying Li, Yuanyuan Li, Feifei Xu, and Xinyu Zhong. 2015. Probabilistic error detecting in numerical linked data. In *International Conference on Database and Expert Systems Applications*. Springer, 61–75.

[13] Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. 2015. Loupe-An Online Tool for Inspecting Datasets in the Linked Data Cloud.. In *International Semantic Web Conference (Posters & Demos)*.

[14] Felix Naumann. 2014. Data Profiling Revisited. *SIGMOD Rec.* 42, 4 (Feb. 2014), 40–49. https://doi.org/10.1145/2590989.2590995

[15] Chifumi Nishioka and Ansgar Scherp. 2016. Information-theoretic Analysis of Entity Dynamics on the Linked Open Data Cloud. In *PROFILES@ ESWC*.

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

[17] Vicky Papavasileiou, Giorgos Flouris, Irini Fundulaki, Dimitris Kotzinos, and Vassilis Christophides. 2013. High-level change detection in RDF (S) KBs. *ACM Transactions on Database Systems (TODS)* 38, 1 (2013), 1.

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

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

[20] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. 2002. Data Quality Assessment. *Commun. ACM* 45, 4 (April 2002), 211–218. https://doi.org/10.1145/505248.506010

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

[22] Giri Kumar Tayi and Donald P Ballou. 1998. Examining data quality. *Commun. ACM* 41, 2 (1998), 54–57.

[23] Raphael Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai-Dominik Kuhn, Catalin Barbu, et al. 2017. 3cixty: Building comprehensive knowledge bases for city exploration. *Web Semantics: Science, Services and Agents on the World Wide Web* (2017).

[24] Fouad Zablith, Grigoris Antoniou, Mathieu d'Aquin, Giorgos Flouris, Haridimos Kondylakis, Enrico Motta, Dimitris Plexousakis, and Marta Sabou. 2015. Ontology evolution: a process-centric survey. *The knowledge engineering review* 30, 1 (2015), 45–75.

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