Quality Assessment of Linked Datasets Using Probabilistic Approximation

Jeremy Debattista(✉), Santiago Londoño, Christoph Lange, and Sören Auer

University of Bonn and Fraunhofer IAIS, Bonn, Germany
{debattis,londono,langec,auer}@cs.uni-bonn.de

Abstract. With the increasing application of Linked Open Data, assessing the quality of datasets by computing quality metrics becomes an issue of crucial importance. For large and evolving datasets, an exact, deterministic computation of the quality metrics is too time consuming or expensive. We employ probabilistic techniques such as Reservoir Sampling, Bloom Filters and Clustering Coefficient estimation for implementing a broad set of data quality metrics in an approximate but sufficiently accurate way. Our implementation is integrated in the comprehensive data quality assessment framework Luzzu. We evaluated its performance and accuracy on Linked Open Datasets of broad relevance.

Keywords: Data quality · Linked data · Probabilistic approximation

1 Introduction

The Web of Data is continuously changing with large volumes of data from different sources being added. Inevitably, this causes the data to suffer from inconsistency, both at a semantic level (contradictions) and at a pragmatic level (ambiguity, inaccuracies), thus creating a lot of noise around the data. It also raises the question of how *authoritative* and *reputable* the data sources are. Taking *DBpedia*[1] as an example, data is extracted from a semi-structured source created in a crowdsourcing effort (i.e. Wikipedia). This extracted data might have quality problems because it is either mapped incorrectly or the information itself is incorrect. *Data consumers* increasingly rely on the Web of Data to accomplish tasks such as performing analytics or building applications that answer end user questions. Information overload is a consistent problem that these consumers face daily. Ensuring quality of the data on the Web is of paramount importance for data consumers, since it is infeasible to filter this infobesity manually.

A particular challenging area is the quality analysis of large-scale, evolving Linked Data datasets. In their editorial [9], Hitzler and Janowicz claim that

This work is supported by the European Commission under the Seventh Framework Program FP7 grant 601043 (http://diachron-fp7.eu).

[1] http://www.dbpedia.org.

F. Gandon et al. (Eds.): ESWC 2015, LNCS 9088, pp. 221–236, 2015.
DOI: 10.1007/978-3-319-18818-8_14

Linked Data is an ideal pilot to experiment with the 4th paradigm of big data (Veracity). However, Linked Data is frequently overlooked due to its reputation of being of poor quality. The quality of data can usually not be described using a single measure, but commonly requires a large variety of quality measures to be computed. Doing this for large datasets poses a substantial data processing challenge. However, for large datasets meticulously exact quality measures are usually not required. Instead users want to obtain an *approximate* indication of the quality they can expect.

Previous work on Linked Data quality analysis primarily employed deterministic algorithms (cf. the survey by Zaveri et al. [23]). Although such algorithms usually have polynomial complexity they are intractable for large datasets and it is difficult to reach runtimes sufficient for practical applications. The rationale of this paper is to show that we can apply probabilistic techniques to assess Linked Data quality. In particular, we employ three techniques commonly used in big data applications: *Reservoir Sampling*, *Bloom Filters* and *Clustering Coefficient estimation*. We develop strategies how these techniques can be applied to boost quality metric computations. We also thoroughly evaluate the quality metrics to tweak the required parameters for more accurate results yet keeping the running time acceptable. All implemented quality metrics are part of a large quality assessment framework, *Luzzu*[2] [5].

The rest of this paper is organised as follows. Section 2 looks at the state of the art. Section 3 provides preliminaries. Section 4 details the Linked Data quality metrics under discussion. Section 5 discusses the implementation of the big data techniques and metrics. Section 6 reports our evaluation results. Final remarks and conclusions are presented in Section 7.

2 State of the Art

In a recent article, Dan O'Brien [17] discusses how big data, which is now being applied in many companies and applications, challenges data governance, including data quality. This section overviews the state of the art in relation to the probabilistic approximation techniques that can be applied to assess data quality in Linked Open Datasets. To our knowledge, there is currently no concrete use of such techniques to assess linked dataset quality.

Since their inception, *Bloom Filters* have been used in different scenarios, including dictionaries and spell-checkers, databases (for faster join operations and keeping track of changes), caching, and other network related scenarios [3]. Recently, this technique was also used to tackle the detection of duplicate data in streams in a variety of scenarios [1,6,11,14]. Such applications included the detection of duplicate clicks on pay-per-click adverts, fraud detection, URI crawling, and identification of distinct users on platforms. Metwally et al. [14] designed a Bloom Filter that applies the "window" principle: sliding windows (finding duplicates related to the last observed part of the stream), landmark windows

[2] Luzzu is open source and available to download from http://eis-bonn.github.io/Luzzu.

(maintaining specific parts of the stream for de-duplication), and jumping windows (a trade-off between the latter two window types). Deng and Rafiei [6] go a step further than [14] and propose the Stable Bloom Filter, guaranteeing good and constant performance of filters over large streams, independent of the streams' size. Bera et al. [1] present a novel algorithm modifying Bloom Filters using *reservoir sampling* techniques, claiming that their approach not only provides a lower *false negative rate* but is also more stable than the method suggested in [6].

Random sampling, in different forms, is often used as an alternative to complex algorithms to provide a quick yet good approximation of results [20]. Sample-based approaches such as the latter were used to assess the quality of Geographic Information System data [18,21]. Xie et al. [21] describe different sampling methods for assessing geographical data. In their approach, Saberi and Ghadiri [18] sampled the original base geographical data periodically. The authors in [12] propose how data quality metrics can be designed to enable (1) the assessment of data quality and (2) analyse the economic consequences after executing data quality metrics. They suggest sampling the dataset attributes to get an estimate measure for the quality of the real-world data.

Lately, various efforts have been made to estimate values within big networks, such as *estimating the clustering coefficient* [8] or calculating the average degree of a network [4]. Hardiman et al. [8] provide an estimator to measure the network size and two clustering coefficient estimators: the network average (local) clustering coefficient and the global clustering coefficient. These measures were applied on public datasets such as DBLP, LiveJournal, Flickr and Orkut. Similarly, Dasgupta et al. [4] calculate the average degree of a network using similar public domain datasets. As Guèret et al. pointed out in [7], network measures can be exploited to assess Linked Data with regard to quality, as Linked Data uses the graph-based RDF data model.

3 Preliminaries

The LOD Cloud[3] comprises datasets having less than 10 K triples, and others having more than 1 billion triples. Deterministically computing quality metrics on these datasets might take from some seconds to days. This section introduces three probabilistic techniques commonly used in big data applications; they combine with a high probability near-to-accurate results with a low running time.

Reservoir Sampling. Reservoir sampling is a statistics-based technique that facilitates the sampling of evenly distributed items. The sampling process randomly selects k elements ($\leq n$) from a source list, possibly of an unknown size n, such that each element in the source list has a k/n probability of being chosen [20]. The reservoir sampling technique is part of the *randomised algorithms* family. Randomised algorithms offer simple and fast solutions for time-consuming counterparts by implementing a degree of randomness. Vitter [20]

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

introduces an algorithm for selecting a random sample of k elements from a bigger list of n elements, in one pass. The author discusses that by using a *rejection-acceptance technique* the running time for the sampling algorithm improves. The main parameter that affects the tradeoff between fast computation and an accurate result is the reservoir size (k). The sample should be *large enough* such that the law of large numbers[4] can be applied.

Bloom Filters. A Bloom Filter [2] is a fast and space efficient bit vector data structure commonly used to query for elements in a set ("is element A in the set?"). The size of the bit vector plays an important role with regard to the precision of the result. A set of hash functions is used to map each item added to be compared, to a corresponding set of bits in the array filter. The main drawback of a Bloom Filter is that they can produce *false positives*, therefore being possible to identify an item as existing in the filter when it is not, but this happens with a very low probability. The trade-off of having a fast computation yet a very close estimate of the result depends on the size of the bit vector. With some modifications, Bloom Filters are useful for detecting duplicates in data streams [1].

Clustering Coefficient Estimation. The clustering coefficient algorithm measures the neighbourhood's density of a node. The clustering coefficient is measured by dividing the number of edges of a node and the number of possible connections the neighbouring nodes can have. The time complexity for this algorithm is $\mathcal{O}(n^3)$, where n is the number of nodes in the network. Hardiman and Katzir [8] present an algorithm that estimates the clustering coefficient of a node in a network using *random walks*. A *random walk* is a process where some object jumps from one connected node to another with some probability of ending in a particular node. A random walker stops when the *mixing time* is reached. In a Markov model, mixing time refers to the time until the chain is close to its steady state distribution, i.e. the total number of steps the random walker should take until it retires. Given the right *mixing time*, the value is proved to be a close approximate of the actual value. The authors' suggested measure computes in $\mathcal{O}(r) + \mathcal{O}(rd_{\max})$ time, where r is the total number of steps in the random walk and $d_{\max}$ is the node with the highest degree[5].

4 Linked Data Metrics

Zaveri et al. present a comprehensive survey [23] of quality metrics for linked open datasets. Most of the quality metrics discussed are *deterministic* and computable within *polynomial time*. On the other hand, once these metrics are exposed to large datasets, the metrics' upper bound grows and as a result, the computational time becomes intractable. In this section we discuss some metrics that are known to suffer from the big data phenomenon.

[4] http://mathworld.wolfram.com/LawofLargeNumbers.html.
[5] The number of in-links plus out-links of a node.

Dereferenceability. HTTP URIs should be dereferenceable, i.e. HTTP clients should be able to retrieve the resources identified by the URI. A typical web URI resource would return a `200 OK` code indicating that a request is successful and a `4xx` or `5xx` code if the request is unsuccessful. In Linked Data, a successful request should return an RDF document containing triples that describe the requested resource. Resources should either be *hash* URIs or respond with a `303 Redirect` code [19]. The dereferenceability metric assesses a dataset by counting the number of valid dereferenceable URIs (according to these LOD principles) divided by the total number of URIs. Yang et al. [22] describe a mechanism[6] to identify the dereferenceability process of a Linked Data resource.

A naïve approach for this metric is to dereference all URI resources appearing in the subject and the object of all triples. In this metric we assume that all predicates are dereferenceable. This means that the metric performs at worst $2n$ HTTP requests, where n is the number of triples. It is not possible to perform such a large number of HTTP requests in an acceptable time.

Existence of Links to External Data Providers. This metric measures the degree to which a resource is linked to external data providers. Ideally, datasets have a high degree of linkage with external data providers, since interlinking is one of the main principles of Linked Data [10].

The simplest approach for this metric is to compare the subject's resource pay-level domain (PLD) against the object's resource PLD[7]. Although this metric is not considered to be computationally expensive ($\mathcal{O}(n)$, where n represents the number of triples), it is also a good candidate for an estimation.

Extensional Conciseness. At the data level, a linked dataset is concise if there are no redundant instances [13]. This metric measures the number of unique instances found in the dataset. The uniqueness of instances is determined from their properties and values. An instance is unique if no other instance (in the same dataset) exists with the same set of properties and corresponding values.

The most straightforward approach is to compare each resource with every other resource in the dataset to check for uniqueness. This gives us a time complexity of $\mathcal{O}(i^2t)$, where i is the number of instances in the datasets and t is the number of triples. The major challenge for this algorithm is the number of triples in a dataset, since each triple (predicate and object) is compared with every other triple streamed from the dataset.

Clustering Coefficient of a Network. The clustering coefficient metric is proposed as part of a set of network measures to assess the quality of data mappings in linked datasets [7]. This metric aims at identifying how well resources are

[6] Also used in the Semantic Web URI Validator Hyperthing (http://www.hyperthing.org).

[7] "PLDs allow us to identify a realm, where a single user or organization is likely to be in control." [16]. For example the PLD for http://dbpedia.org/resource/Malta is dbpedia.org.

Table 1. Mapping probabilistic approximation techniques with Linked Data quality metrics

Probabilistic Approximation Technique	Linked Data Metric
Reservoir Sampling	Dereferenceability
	Links to External Data Providers
Bloom Filters	Extensional Conciseness
Clustering Coefficient Estimation	Clustering Coefficient of a Network

connected, by measuring the density of the resource neighbourhood. A network has a high clustering cohesion when a node has a large number of neighbouring nodes, all of which are connected to each other. This means that links may end up being meaningless [7].

When assessing the clustering coefficient of a network, a graph is built where the *subject* and *object* of a triple (either URI resources or blank nodes) are represented as vertices in the graph, whilst the *predicate* is the edge between them. As this ignores triples with literal objects, there is no direct correlation between the number of triples in a dataset and number of vertices. Calculating this measure on a network takes $\mathcal{O}(n^3)$, especially for large datasets. This is because each vertex in the network has to be considered: for each vertex v in the graph, we identify the number of links between the neighbours of v (i.e. how many of v's neighbours are connected together) and divide it by the number of possible links.

5 Implementation

Based on the probabilistic techniques described in Sect. 3, we analyse how they can help in assessing quality in linked datasets. These metrics are implemented as an extensible package for *Luzzu*. Luzzu [5] is a Linked Data quality assessment framework that provides an integrated platform which: (1) assesses Linked Data quality using a library of generic and user-provided domain specific quality metrics in a scalable manner; (2) adds queryable quality metadata to the assessed datasets; and (3) assembles detailed quality reports on assessed datasets. Datasets are assessed using a sequential streaming approach. Table 1 shows which approximation can be used for each respective metric.

5.1 Reservoir Sampling

Our implementation is based on the *rejection-acceptance* technique [20]. The trade-off parameter is the definition of the maximum number of items (k) that can be stored. Various factors are taken to define k, such as the rough estimation of the size of the dataset and available memory, since this reservoir is stored in-memory.

When attempting to add an *item* to the reservoir sampler, an item counter (n) is incremented. This increment is required to calculate the *replacement probability*, since the exact size of the source (in our case the dataset) is unknown. The *item* can be (i) *added* to the reservoir, (ii) become a *candidate* to replace another item, or (iii) be *discarded*. The first possible operation is straightforward. If the reservoir sampler has free locations ($n < k$), the *item* is added. On the other hand, when the reservoir is full, the *item* can either replace another item in the list, or rejected. The decision is made by generating a random number (p) between 0 and n. If p lies in the range of the reservoir list length (i.e. $p < k$), then the new *item* replaces the current item stored in that position of the reservoir, else it is rejected. This simulates the k/n *replacement probability* for all items.

Estimated Dereferenceability Metric. Each resource URI is split into two parts: (1) the pay-level Domain (PLD), and (2) the path to the resource. For this metric we employ a "global" reservoir sampler for the PLDs. Furthermore, for each PLD we employ another reservoir sampler holding an evenly distributed sample list of resources to be dereferenced. If the pay-level domain returns a `4xx/5xx` code upon an HTTP request, then all other sampled resources in that reservoir are automatically deemed as non-dereferenceable. Envisaging the possibility of multiple HTTP requests to same domain or resource, we make use of the Luzzu's caching mechanism, to store HTTP requests. The metric value is calculated as a ratio of the total number of dereferenced URIs against the total number of sampled URIs.

Estimated Links to External Data Providers Metric. In order to measure the use of external data providers, the metric must first identify the base URI of the dataset that is being assessed. As each triple is streamed to the metric processor, a heuristic mechanism identifies the base URI. For this, we apply one of the two heuristics, listed in order of priority:

1. Extract the base URI from a triple having the *predicate* `rdf:type` and *object* `void:Dataset` or `owl:Ontology`.
2. The URI (PLD) with the maximum number of occurrences in the *subject* of the assessed dataset.

Each triple's *object* in the dataset is then used to estimate the value of this metric, by first extracting its PLD and attempting to add it to the metric's reservoir. The value of this metric is defined as the ratio of the number of PLDs in the sampler that are not the same as the base URI, against the total number of URIs in the sampler.

5.2 Bloom Filters

Linked datasets might suffer from instance duplication. Bera et al. [1] introduced some modifications to the mechanics of Bloom Filters to enable the detection

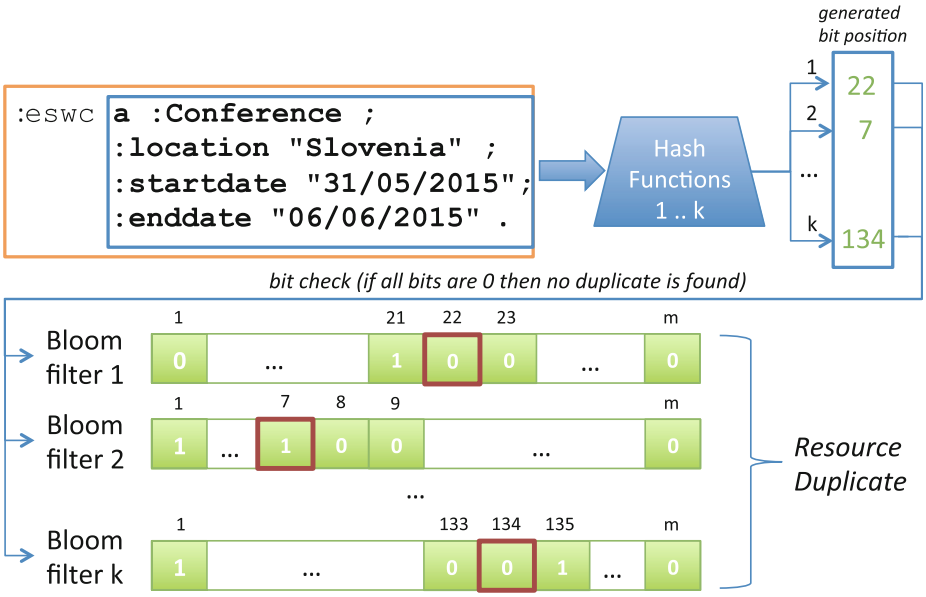

Fig. 1. Illustrating bloom filters with an example

of duplicate elements in data streams. These modifications allow items to be inserted indefinitely by probabilistically resetting bits in the filter arrays when they are close to getting overloaded. The Randomised Load Balanced Biased Sampling based Bloom Filter (RLBSBF) is used to implement the detection of duplicate instances. The authors show that this approach is efficient and generates a low *false positive* rate.

An RLBSBF algorithm is initialised with (1) the total memory used by filter arrays in bits (M); and (2) a threshold value (t_{FPR}) for the false positive rate. The bit vector is initialised with k Bloom Filters. Each bloom filter has a size of M/k and a hash function is mapped to it. The authors in [1] suggest that k is calculated using the threshold value t_{FPR}. A high threshold value means faster computation but less accurate results.

Whenever a new element is processed, the Bloom Filter sets all k bit positions using the hash functions mapped to them. If the bit positions were previously set in the bit vector, it means that a duplicate was detected. Otherwise, the probabilistic resetting of bits is performed before the new element is added to the bit vector. Our implementation uses 128-bit Murmur3[8] hashing functions. Figure 1 illustrates how Bloom Filters help to identify a Linked Data resource that already exists in a dataset.

Estimated Extensional Conciseness Metric. When triples are streamed to the metric processor, the *predicate* and *object* are extracted and serialised as a

[8] https://code.google.com/p/smhasher/wiki/MurmurHash3.

string. The latter string is stored in a sorted set. This process is repeated until a triple with a different *subject* identifier is processed. The sorted set is then flattened to a string and added to the Bloom Filter, discovering any possible duplicates. The set is then initialised again for the new resource identifier and the process is repeated until no more triples are streamed.

The main drawback of our proposed algorithm is that a dataset must be sorted by *subject*, such that all triples pertaining to the same instance are streamed one after another. Although it is common practice to publish datasets sorted by subject (e.g. DBpedia), this cannot be guaranteed in the general case. In our experiments we pre-process RDF dumps by converting them to the N-Triples serialisation, which can be sorted by subject in a straightforward way.

5.3 Clustering Coefficient Estimation

In [8], the authors propose an approach for estimating a social network's clustering coefficient by creating a random walk. In their proposed algorithm, Hardiman and Katzir use $\log^2 n$[9] as the base *mixing time*, i.e. the number of steps a random walker takes until it converges to a steady-state distribution. However, different network characteristics lead to different *mixing time*s, where well-connected networks have a small (fast) mixing time [15].

To calculate an estimate of the clustering coefficient given a random walk $R = \{x_1, x_2, \ldots, x_r\}$, Hardiman and Katzir propose the Estimator 1:

Estimator 1

$$\Phi_l = \frac{1}{r-2} \sum_{r-1}^{k=2} \phi_k \frac{1}{d_{x_k} - 1}$$

$$\Psi_l = \frac{1}{r} \sum_{r}^{k=1} \frac{1}{d_{x_k}}$$

$$\hat{c}_l \triangleq \frac{\Phi_l}{\Psi_l}$$

where r is the total number of steps in the random walk R, x_k is the index of the k^{th} node in the random walk, d_{x_k} is the degree of node x_k and ϕ_k represents the value in the adjacency matrix A in position $A_{x_{k-1}, x_{k+1}}$.

Estimated Clustering Coefficient Metric. When triples are streamed into the metric, the vertices are created by extracting the *subject* and the *object*, whilst the *predicate* acts as a directed edge between the two nodes. We use URI resources and blank nodes to create the network vertices. To calculate the estimated clustering coefficient value, a random walk is performed on the graph.

[9] The square of $\log n$.

Similarly to the approach in [8], we view the graph as undirected. The idea is that if the random walker ends up in a dead-end (i.e. cannot move forward), it can go back to continue crawling the network. Our mixing time parameter is $m \log^2 n$. Since linked open data advocates interlinking and re-use of resources, we expect[10] that such datasets have a low mixing time. The multiplier factor m thus enables us to increase or decrease the mixing time as required. The reason behind this is to enable a parameter modifier to the base mixing time ($\log^2 n$), since it is difficult to find a one size fits all mixing time. Estimator 1 is used to obtain a close estimate of the dataset's clustering coefficient. Finally, the estimated value is normalised as described in [7].

6 Metric Analysis and Experiments

Having implemented the metrics using probabilistic approximation techniques, we measure the computed quality metric values and runtime for the approximate metrics and compare them with the actual metrics. For each approximate metric, we experimented with different parameter settings to identify the best parameter values. All tests are run on a Unix virtual machine with an Intel Xeon 3.00 GHz, with 3 cores and a total memory of 3.8 GB. We used a number of datasets of varying sizes and covering different application domains. We found them on Datahub, looking for datasets tagged with the *lod* tag. These are:

- Learning Analytics and Knowledge (LAK) Dataset ≈ 75 K triples;
- Lower Layer Super Output Areas (LSOA) ≈ 280 K triples;
- Southampton ECS E-Prints Dataset ≈ 1 M triples;
- WordNet 2.0 (W3C) Dataset ≈ 2 M triples;
- Sweto DBLP Dataset ≈ 15 M triples;
- Semantic XBRL ≈ 100 M triples;

Parameter Setting. In order to maximise accuracy, the parameters of the algorithms have to be tweaked. Therefore, we experimented with different parameter values and analysed the metric results. Parameter settings were obtained by observing the algorithm's parameters in correlation with the datasets and metrics. The rationale behind this experiment is to identify a single parameter that, when used in a metric, gives acceptable results within reasonable time. This experiment was not performed on all datasets, since in certain cases the actual metric does not complete its computation.

The *Dereferenceability* metric was implemented using reservoir sampling. Table 2 shows the time taken (in seconds) and the approximate value for different parameter settings. The biggest time factor in this metric is the network access time, i.e. the time an HTTP request takes to respond. The parameter settings employed for this experiment are: (P1) global reservoir size: 10, PLD reservoir size: 1000; (P2) global reservoir size: 50, PLD reservoir size: 100; (P3) global

[10] We are currently performing research on the mixing time of the linked datasets available in the LOD Cloud.

Table 2. Dereferenceability metric with different parameter settings

	Time (s)	Value		Time (s)	Value
Actual (LAK)	1423.33	0.045533	**Actual (LSOA)**	3189.819	0
P1	842.082	0.345548	P1	149.913	0
P2	426.892	0.162581	P2	80.748	0
P3	618.761	0.057866	P3	388.522	0
P4	480.972	0.262936	P4	316.862	0

Table 3. Existence of links to external data providers metric with different parameter settings

	Time(s)	Value		Time(s)	Value
Actual (LAK)	2.271	0.000156	**Actual (LSOA)**	3.529	3.272144×10^6
P1	0.578	0.000156	P1	1.287	3.272144×10^6
P2	0.481	0.000156	P2	1.182	3.272144×10^6
P3	0.466	0.000156	P3	1.153	3.272144×10^6
P4	0.444	0.000156	P4	1.124	3.272144×10^6

	Time(s)	Value		Time(s)	Value
Actual (S'OTON)	23.872	6.166189×10^6	**Actual (WN)**	33.82	0
P1	20.693	6.166189×10^6	P1	7.362	0
P2	20.008	6.166189×10^6	P2	7.779	0
P3	20.4	6.166189×10^6	P3	7.557	0
P4	20.589	6.166189×10^6	P4	7.258	0

reservoir size: 50, PLD reservoir size: 10000; (P4) global reservoir size: 100, PLD reservoir size: 1000. Whilst the approximate metrics completed the computation for all datasets, the exact computation was only ready for the LAK and LSOA datasets. Based on the available results from the datasets, we can conclude that the optimal parameter for this metric is close to the P3 settings. The results for the LSOA dataset are 0 due to the fact that all resources returned a `4xx/5xx` error. This was verified manually.

Another application of the Reservoir Sampling was the *Existence of Links to External Data Providers*. Table 3 shows the time taken (in seconds) and the estimated value for different parameter settings. The parameter settings used to initialise the sampler were: (P1) 5,000; (P2) 10,000; (P3) 20,000; (P4) 50,000. The results show that the approximation technique did not record any major difference up to 2 M, but the technique fares better with very big datasets (c.f. Fig. 2). One possible reason for this is that since the actual metric is not expected to fit in-memory, our implementation uses *MapDB*[11], a pure Java database that stores memory data structures such as hash maps on disk. It is also worth noting that all estimates gave the same value as the actual. The reason for this is that the number of object PLDs fits in the smallest reservoir. Therefore, since the runtime between different parameters varies a little, setting a higher or lower reservoir sampler in this case is a matter of available memory space. If all PLDs fit in the reservoir sampler, the result is 100 % accurate.

[11] http://www.mapdb.org.

Table 4. Extensional conciseness metric with different parameter settings

	Time(s)	Value
Actual (LAK)	81.334	0.994860
P1	1.348	0.621315
P2	1.377	0.962249
P3	1.67	0.993946
P4	2.212	0.994593

	Time(s)	Value
Actual (LSOA)	375.873	1
P1	1.043	0.617729
P2	1.328	0.966795
P3	1.807	0.999240
P4	2.98	1

	Time(s)	Value
Actual (S'OTON)	7366.225	0.737523
P1	24.304	0.512887
P2	20.217	0.782946
P3	17.512	0.783529
P4	20.275	0.660193

	Time(s)	Value
Actual (WN)	96511.334	0.948
P1	7.407	0.570991
P2	11.502	0.885790
P3	17.653	0.900407
P4	35.381	0.844733

Table 5. Clustering coefficient metric with different parameter settings

	Time(s)	Value
Actual (LAK)	42.729	0.961040
Mixing time 0.1	4.595	0.978220
Mixing time 0.5	4.595	0.997945
Mixing time 0.7	4.766	0.998665
Mixing time 1.0	4.832	0.998974

	Time(s)	Value
Actual (LSOA)	62.618	1
Mixing time 0.1	7.657	0.999995
Mixing time 0.5	6.829	0.999999
Mixing time 0.7	6.561	0.999503
Mixing time 1.0	6.528	0.999999

	Time(s)	Value
Actual (S'OTON)	408.358	0.933590
Mixing time 0.1	46.373	0.993067
Mixing time 0.5	46.362	0.997634
Mixing time 0.7	46.238	0.997939
Mixing time 1.0	46.225	0.998312

	Time(s)	Value
Actual (WN)	9012.454	0.759257
Mixing time 0.1	243.009	0.810405
Mixing time 0.5	248.925	0.999919
Mixing time 0.7	251.396	0.999917
Mixing time 1.0	252.522	0.999967

The *Extensional Conciseness* metric was implemented using Bloom Filters. Table 4 shows the time taken (in seconds) and the estimated value for different parameter settings. We applied 4 different settings for experimentation: (P1) 2 filters (k) with a size (M) of 1,000; (P2) 5 filters with a size of 10,000; (P3) 10 filters with a size of 100,000; (P4) 15 filters with a size of 10,000,000. This technique showed a lot of potential in the de-duplication process. The time taken in the approximate algorithms are lower than the actual, with results being almost as accurate. Based on the Bloom Filter trade-off, a setting between P3–P4 would exploit the potential of this technique in assessing the quality of linked datasets with regard to duplication problems.

For the *clustering coefficient* metric we multiplied the base mixing time of $\log^2 n$ with 0.1, 0.5, 0.7 and 1.0 respectively to test with fast mixing time. Table 5 shows the time taken (in seconds) and the estimated value for different parameter settings. The results show that for the assessed datasets the $\log^2 n$ mixing time is not ideal. This is due to the fact that the smallest multiplier setting, i.e. 0.1, proved to be the closest to the actual result in all cases. Determining a more accurate average mixing time, and hence a more accurate estimate (cf. Sect. 3), requires the evaluation (such as in [15]) of all datasets in the LOD Cloud.

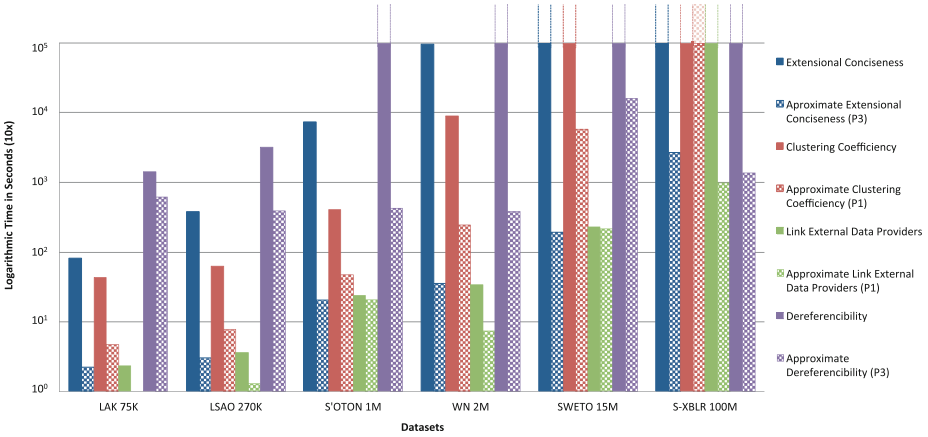

Fig. 2. Runtime of Metrics vs. Datasets

Table 6. Metric value (actual and approximate) per dataset

	LAK 75K	LSOA 270K	S'OTON 1M	WN 2M	SWETO 15M	S-XBLR 100M
Extensional Conciseness	0.9948	1	0.7375	0.948	0,000370	N/A
Approx. Extensional Conciseness	0.9945	1	0.6601	0.8447	0.9998	0.1097
Clustering Coefficiency	0.9610	1	0.9335	0.7592	N/A	N/A
Approx. Clustering Coefficiency	0.9782	0.9999	0.9930	0.8104	1	0
Link External Data Providers	0.01569×10^{-6}	3.2721×10^{-6}	6.1661×10^{-6}	0	N/A	N/A
Approx. Link External Data Prov.	0.01569×10^{-6}	3.2721×10^{-6}	6.1661×10^{-6}	0	0.000370	$4{,}9557\times10^{-8}$
Dereferencibility	0.0455	0	N/A	N/A	N/A	N/A
Approx. Dereferencibility	0.0578	0	0.4122	0.9681	0	$0{,}0955\times10^{-8}$

Evaluation Discussion. Our experiments gave promising results towards the use and acceptance of probabilistic approximation for estimating the quality of linked open datasets. Figure 2 shows the time taken in all implemented metrics (actual and approximated) against the evaluated datasets. The graph clearly shows that all approximate metrics have a lower runtime than their equivalent actual metric. Whilst the approximate metrics for *link external data providers*, *dereferenceability*, and *extensional conciseness* computed all metrics, the approximate *clustering coefficient* and the actual *link external data providers* managed to compute 5 datasets within a reasonable time. The actual *dereferenceability*

Table 7. Possible metric approximation implementation

Metric	Approximation technique
Dereferenceability of the URI	Reservoir Sampling
Dereferenced Forward-Links	Reservoir Sampling
Detection of Good Quality Interlinks	Random Walk
Dereferenced Back-Links	Reservoir Sampling
Usage of Slash-URIs	Reservoir Sampling
Syntactically Accurate Values	Reservoir Sampling
No Misuse of Properties	Reservoir Sampling
No Use of Entities as Members of Disjoint Classes	Reservoir Sampling
High Extensional Conciseness	Bloom Filters
High Intensional Conciseness	Bloom Filters
Duplicate Instance	Bloom Filters
Relevant Terms Within Meta-Information Attributes	Page Rank
Coverage	Reservoir Sampling

metric managed only to compute two datasets, while the other two actual metrics computed up to the WordNet dataset. Table 6 shows the metric (actual and estimated) values for the datasets. The approximate results are in most cases very close to the actual results. However, approximate measures are calculated in an acceptable time unlike their actual counterparts. As part of a larger effort to implement scalable LOD quality assessment metrics, we assessed the metrics identified in [23] and assigned to them possible approximation techniques discussed in this article (cf. Table 7).

Overall, given that the results were obtained on yet small datasets (the chosen ones might not be considered to be big enough) due to limited infrastructure, this paper contributes towards invaluable results that can be the basis for further studies. These results show that with probabilistic approximation techniques:

1. Runtime decreases considerably – for larger datasets easily by more than an order of magnitude;
2. Loss of precision is acceptable in most cases with less than 10% deviation from actual values;
3. Large linked datasets can be assessed for quality even within very limited computational capabilities, such as a personal notebook.

7 Conclusion

In this article, we have demonstrated how the three approximate techniques reservoir sampling, Bloom Filters and clustering coefficient estimation can be successfully applied for Linked Data quality assessment. Our comprehensive experiments have shown that we can reduce runtime in most cases by more

than an order of magnitude, while keeping the precision of results reasonable for most practical applications. All in all, we have demonstrated that using these approximation techniques enables data publishers to assess their datasets in a convenient and efficient manner without the need of having a large infrastructure for computing quality metrics. Therefore, data publishers are encouraged to assess their data before publishing it to the Web, thus ensuring that data consumers receive quality data at their end.

In terms of Linked Data quality assessment we aim to extend our work both in terms of used big data techniques and metric coverage. Regarding probabilistic approximation techniques, we aim to assess other probabilistic data structures such as quotient filters or random trees. A further interesting avenue of research is to investigate how such techniques can be easily employed for domain specific data quality metrics.

References

1. Bera, S.K., Dutta, S., Narang, A., Bhattacherjee, S.: Advanced Bloom filter based algorithms for efficient approximate data de-duplication in streams (2012)
2. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
3. Broder, A.Z., Mitzenmacher, M.: Network applications of Bloom filters: a survey. Internet Math. **1**, 485–509 (2004)
4. Dasgupta, A., Kumar, R., Sarlos, T.: On estimating the average degree. In: WWW, pp. 795–806. ACM, New York (2014)
5. Debattista, J., Londoño, S., Lange, C., Auer, S.: LUZZU - a framework for linked data quality assessment (2014). http://arxiv.org/abs/1412.3750
6. Deng, F., Rafiei, D.: Approximately detecting duplicates for streaming data using stable bloom filters. In: ACM SIGMOD 2006, pp. 25–36. ACM (2006)
7. Guéret, C., Groth, P., Stadler, C., Lehmann, J.: Assessing linked data mappings using network measures. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 87–102. Springer, Heidelberg (2012)
8. Hardiman, S.J., Katzir, L.: Estimating clustering coefficients and size of social networks via random walk. In: WWW, pp. 539–550. ACM (2013)
9. Hitzler, P., Janowicz, K.: Linked data, big data, and the 4th paradigm. Semant. Web **4**(3), 233–235 (2013)
10. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An empirical survey of linked data conformance. J. Web Sem. **14**, 14–44 (2012)
11. Jain, N., Dahlin, M., Tewari, R.: Taper: tiered approach for eliminating redundancy in replica synchronization. In: FAST, USENIX (2005)
12. Kaiser, M., Klier, M., Heinrich, B.: How to measure data quality? - a metric-based approach. In: International Conference on Information Systems (ICIS), p. 108 (2007)
13. Mendes, P.N., Mühleisen, H., Bizer, C.: Sieve: linked data quality assessment and fusion. In: 2012 Joint EDBT/ICDT Workshops, pp. 116–123. ACM (2012)
14. Metwally, A., Agrawal, D., Abbadi, A.E.: Duplicate detection in click streams. In: WWW 2005. ACM (2005)

15. Mohaisen, A., Yun, A., Kim, Y.: Measuring the mixing time of social graphs. In: SIGCOMM. IMC 2010, pp. 383–389. ACM (2010)
16. Mühleisen, H.: Vocabulary usage by pay-level domain (2014). http://webdatacommons.org/structureddata/vocabulary-usage-analysis/
17. O'Brien, D.: January 2015. http://insideanalysis.com/2015/01/the-key-to-quality-big-data-analytics/
18. Saberi, B., Ghadiri, N.: A sample-based approach to data quality assessment in spatial databases with application to mobile trajectory nearest-neighbor search (2014)
19. Sauermann, L., Cyganiak, R.: Cool URIs for the semantic web. Interest Group Note, W3C, December 2008. http://www.w3.org/TR/2008/NOTE-cooluris-20081203/
20. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. **11**, 37–57 (1985)
21. Xie, H., Tong, X.H., Jiang, Z.Q.: The quality assessment and sampling model for the geological spatial data in China. In: ISPRS Archives - Volume XXXVII, Part B2. ISPRS 2008, pp. 819–824 (2008)
22. Yang, Y., Che, H., Gibbins, N., Hall, W., Shadbolt, N.: Dereferencing semantic web URIs: what is 200 OK on the semantic web?. https://dl.dropboxusercontent.com/u/4138729/paper/dereference_iswc2011.pdf
23. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment methodologies for linked open data. Semant. Web J. (2014). http://www.semantic-web-journal.net/content/quality-assessment-linked-data-survey