Improving Linked Data Quality using Outlier Detection

Jeremy Debattista
University of Bonn / Fraunhofer IAIS
Bonn, Germany
debattis@cs.uni-bonn.de

Christoph Lange
University of Bonn / Fraunhofer IAIS, Germany
Bonn, Germany
langec@cs.uni-bonn.de

Sören Auer
University of Bonn / Fraunhofer IAIS, Germany
Bonn, Germany
auer@cs.uni-bonn.de

ABSTRACT

With more and more data being published on the Web as Linked Data, Web Data quality is increasingly important. While quite some work has been done with regard to quality assessment of Linked Data, only few works have addressed quality improvement. In this article, we present an approach for identifying potentially incorrect RDF statements. The approach is based on distance-based k-nearest neighbour (k-NN) style clustering of statements with the same property and identifying outliers from these clusters. Our method follows a three stage approach, which automates the whole process of finding potentially incorrect statements for a certain property. In the *initial* stage, RDF statements are added to a reservoir sampler based on Vitter's *rejection-acceptance* technique. The *mapping* stage groups data objects in various cells. Finally, the *colouring* stage identifies the cells that contain outlier data objects. The proposed approach is scalable in terms of its polynomial *time complexity*, and also in terms of its *space complexity*. Our detailed empirical evaluation shows that a high precision is maintained with different settings, which effectively facilitates improving the quality of datasets at a large scale.

Categories and Subject Descriptors

The Web of Data [**Vocabularies, taxonomies and schemas for the Web of Data**]

General Terms

Experimentation, Measurement, Reliability

1. INTRODUCTION

A rationale of the Semantic Web is to provide real-world things, also called *resources*, with descriptions in common data formats that are meaningful to machines. Furthermore, Linked Data emphasizes on the reuse and linking of these resources, thus assisting in the growth of the *Web of* (meaningful) *Data*. Schemas, some being lightweight and others being more complex, have been defined for various use cases and application scenarios in order provide structure to the descriptions of semantic resource based on a common understanding. Nevertheless, since linked datasets are usually originating from various structured (e.g. relational databases), semi-structured (e.g. Wikipedia) or unstructured sources (e.g. plain text), a complete and accurate *semantic lifting* process is difficult to attain. Such processes often contribute to incomplete, misrepresented and noisy data, especially for semi-structured and unstructured sources. Issues caused by these processes can be attributed to the fact that either the knowledge worker is not aware of the various implications of a schema (e.g. incorrectly using inverse functional properties), or because the schema is not well defined (e.g. having an open domain and range for a property). In this article, we are concerned with the latter cause, and aim to identify potentially incorrect statements in order to improve the quality of the knowledge base.

As an example, when analysing the schema of the DBpedia dataset, we found out that, from around 61,000 properties, approximately 59,000 had an undefined domain and range. This means that the type of resources attached to such properties as the subject or the object of an RDF triple can be very generic, i.e. `owl:Thing`. For example, the property `dbp:author`[1], whose domain and range are undefined, has instances where the subject is of type `dbo:Book` and the object of type `dbo:Writer`, and other instances where the subject is of type `dbo:Software` and the object of type `dbo:ArtificialSatellite` (<dbpedia:Cubesat_Space_Protocol> <dbp:author> <dbpedia:AAUSAT3>).

Without looking at a schema one would intuitively expect that some *Work* has a *Person* as its *author*, but if the *author* property is underspecified, its semantics cannot be fully understood. Thus, lifting such domain and range restrictions on properties can undeniably increase the possibility of having incorrect RDF statements. Such statements decrease the quality of the dataset and as a result the precision of results when querying, in turn affecting *data consumers* who use the knowledge base, including service providers as well as end users.

The aim of this research is to determine whether potentially incorrect RDF statements can be detected using a clustering technique, based on the subject and the object type of a statement's triple. A key aspect of this technique is to identify a semantic similarity measure that increases the precision of results, as it is required in order to group similar triples in clusters. For example, two statements having a subject of type *book* and an object of type *writer* and another object of type *soccer manager* should be clustered together, as soccer managers may also be writers of books. The goal

[1]Unless otherwise stated, prefixes used are as defined in `http://www.prefix.cc`.

is that with the proposed technique we can identify incorrect statements in acceptable runtime, which can then be removed from the knowledge base, thus improving its quality.

Our method follows a three stage approach, which automates the whole process of finding potentially incorrect statements for a certain property. In the *initial* stage, RDF statements are added to a reservoir sampler based on Vitter's *rejection-acceptance* technique. The *mapping* stage groups data objects in cells in a two-dimensional grid. Finally, the *colouring* stage identifies the cells that contain outlier data objects. This three stage approach computes in polynomial time, whilst the data structures used are fast and space efficient.

We evaluate complementary aspects of the proposed approach. More specifically, we were interested to see how different settings in our approach affect the precision and recall values, and if the approximative quality value varies from the actual quality value by a large degree. For the former we observe that, while we get high precision values, in most cases over 0.8, the recall values are mostly around 0.4. With regard to the latter, we observe that the approximate quality value, in two use cases, varies by $\pm$ 0.1.

Overall the particular contributions of this work are:

- the definition of an outlier detection method aiming at effectively (high precision) and efficiently (scalability) detecting statements that are potentially incorrect w.r.t. domain and range,
- the detailed analysis of the method with regard to various similarity measures, parameters as well as regarding its time and space complexity.

Most existing work in Linked Data quality improvement focuses on detecting potentially incorrect literal values through, e.g., statistical analysis [14], crowdsourcing [13, 1] or using background knowledge for schema enrichment [15, 11]. Little work has been done to detect possible quality problems of object properties (e.g. [10] employing the statistical distribution of types over properties). Hence, a particular innovation of our approach is to take into consideration the semantic topology of types and not just their statistical usage.

As a result of applying outlier detection to detect possible incorrect statements as described in this article and its subsequent cleaning, the quality of Linked Data can be substantially improved. The main asset of our method is that the usage barrier is very low, since no initial supervision, configuration or training is required. Our method generates potential quality problems immediately and the high precision values make the manual review of potential quality problems relatively efficient. However, the recall we achieved still leaves room for further improvements in subsequent work. Also, as we show with a detailed time and space requirements analysis, our algorithm has good scaling properties and is thus capable of processing big datasets. Our implementation of the method was integrated into the comprehensive open-source Linked Data Quality assessment framework *Luzzu* and is, together with all the experimental analysis results, available from `http://eis-bonn.github.io/Luzzu/`.

This article is structured as follows. Section 2 describes the background techniques required for this article. Our proposed approach is explained in Section 3, together with the analysis of its space and time complexity. Experiments and evaluations of our approach are documented in Section 4. The state of the art is described in Section 5, whilst conclusions and an outlook to future work are discussed in Section 6.

2. PRELIMINARIES

In this section we introduce the techniques used in our approach to detect potentially incorrect RDF statements in Linked Data. Our idea is to make use of distance-based outlier detection techniques, more specifically to apply unsupervised clustering to statements. We apply the technique defined by Knorr et al. in [8], with some modifications to improve computation time (using indexes) and automation (using reservoir sampling to find initial values). Specific details on our approach are discussed in Section 3.

2.1 Distance-Based Outlier Detection

Outlier detection is utilised in numerous applications such as fraud detection and text originality detection. In [7], Hodge and Austin describe three different types of approaches for outlier detection. The *Type 1* approach determines outliers "with no prior knowledge of the data" [7]. Assuming a normal distribution, the type 1 approach separates data that looks normal from outliers. This approach is usually suited for statistical approaches with *uni*variate data following a known distribution (e.g. normal or Gamma distribution). However, in our case, the data objects (i.e. RDF statements) are *two*-dimensional in relation to the statement's property, as every statement has a subject and an object.

Knorr et al. [8] propose a distance-based technique to overcome this barrier using a k-nearest neighbour (k-NN) style clustering. The authors state that given a maximum number M of allowed objects and a distance D, outliers can be detected by searching for objects within the radius D of an object O. The rationale is that the data is partitioned into cells in a two-dimensional space, according to the following axioms:

1. Two data objects are in the same cell **iff** their distance is $D/2$;
2. Two data objects are in two different but encircling or buffering cells **iff** their distance is at most D. *Encircling* cells (described as $Layer_1$ in [8]) are those cells surrounding the host cell with coordinates $x \pm 1$ and $y \pm 1$. *Buffer* cells (described as $Layer_2$ in [8]) are those cells surrounding the host cell with coordinates $x \pm 3$ and $y \pm 3$;
3. Two data objects are at least three cells apart from each other **iff** their distance is at least D.

Figure 1 depicts a two-dimensional space with various data objects (marked as black dots) in cells. A cell which has $M+1$ data objects is coloured *red*, whilst its encircling cells are coloured *pink*. This is because the objects in the *red* cell and the objects in the *pink* cells are within distance D of each other. If two or more adjacent cells have more than $M + 1$ data objects together, but less then $M + 1$ individually, are coloured *pink* and thus not marked as outliers.

Hodge and Austin [7] state that such techniques suffer from exponential complexity, though Knorr et al. [8] suggest a linear running time of $\mathcal{O}(m + N)$, where m is the number of cells and N is the number of data objects in a dataset. In our approach we reduce the runtime further by using indexes.

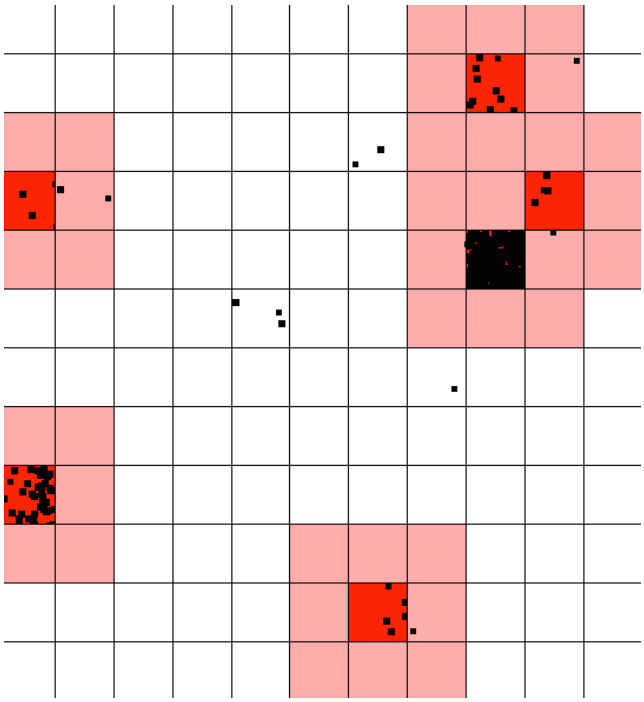

Figure 1: A graphical representation of distance-based outlier detection according to Knorr et al. [8] using a fictitious sample set of RDF statements, and cell colouring.

2.2 Reservoir sampling

Reservoir sampling is a statistics-based technique that facilitates the sampling of evenly distributed items. The sampling process randomly selects k elements ($\leq c$ – where c is a counter of attempted additions) from a source list, possibly of an unknown size c, such that each element in the source list has a k/c probability of being chosen [12]. The reservoir sampling technique is part of the *randomised algorithms* family. Randomised algorithms offer simple and fast improvements to their time-consuming counterparts by implementing a degree of randomness while still computing correct results with a high probability. Vitter [12] introduces an algorithm for selecting a random sample of k elements from a bigger list of c 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 can be applied.

2.3 Semantic Similarity Measures

Semantic similarity measures are metrics that are used to determine the semantic relationship between the characteristics of two elements. These measures consider the underlying schema of the instances, more specifically the '*is a*' relations, in order to determine a similarity value between two concepts. More specifically, Harispe et al. [6] defined the notion of semantic measure as a "theoretical tool or function which enables the comparison of elements according to semantic evidences". In order to clarify this statement, let us consider, for example, the concept *writer* and the concept *comic creator*. These two concepts are **similar** to each other as they are both *persons*, but using such similarity measures we cannot say that they are related to the *book* concept.

In their survey, Harispe et al. [6] describe two different types of semantic evidence: *extensional* and *intentional*. The former is based on the analysis of the topology and usage of classes (i.e. the number of instances associated to that class) and thus the measure is biased towards the usage. This is also known as extrinsic information content. On the other hand, the intentional evidence (or intrinsic information content) considers only the topology of the classes. Although these kind of measures preclude the usage bias, it is assumed that the schema is manifested in depth such that enough knowledge is available to calculate the semantic measure of two concepts. Our generic approach allows for both kind of measures.

3. DETECTING INCORRECT STATEMENTS

The detection and subsequent cleaning of potentially incorrect RDF statements aids in improving the quality of a linked dataset. There were a number of attempts to solve this problem in the best possible manner (cf. Section 5). More recently, Paulheim and Bizer presented an approach whereby a statistical distribution measure was used to validate the correctness of RDF statements [10]. Similarly, our approach focuses on detecting incorrect statements where both the subject and object are resources. We apply the techniques described in Section 2, more specifically the distance-based outlier technique by Knorr et al. [8] in a Linked Data scenario. Exploiting reservoir sampling and semantic similarity measures, clusters of RDF statements (having the same property but different subject and object resources) are created, thus identifying the potentially incorrect statements. As opposed to [10], our approach focuses on finding incorrect statements using outlier detection rather than using a statistical frequency (distribution) of types. We have also also implemented this approach as a metric for our *Luzzu* linked data quality assessment framework [2].

3.1 Approach

Our proposed method[2] has three stages: *initial*, *mapping*, and *colouring*. These three stages automate the whole process of finding potentially incorrect statements for a certain property. In the *initial* stage, k (the size of the reservoir) RDF statements are added to a reservoir sampler based on Vitter's *rejection-acceptance* technique [12]. Following the initialisations of the constants, the *mapping* stage groups data objects in various cells based on the properties described in Section 2.1. Finally, the *colouring* stage identifies the cells that contain outlier data objects.

3.1.1 Initial Stage

The initial steps are crucial for achieving a more accurate result, i.e. a better identification of potentially incorrect statements. We start by determining the approximate distance D that is used in the second stage to condition the mapping, and thus the final clustering, of RDF statements. The approximate value D is valid for a single property, i.e. the property whose triples are being assessed. Therefore, two properties (e.g. *dbp:author* and *dbp:saint*, i.e. the patron saint of, e.g., a town) will have different values of D according to the triples and their types. Currently, in our approach we assume that for resource only one type is returned (referring to the function `typeOf` in 1), i.e. the most specific type assigned to the

[2]The implementation of our approach can be found at `https://github.com/diachron/quality`

resource (e.g. *dbo:Writer* and not *dbo:Person*). A threshold fraction p, between 0 and 1, which is defined by the user (the only non-automated part of this approach), conditions the number of data objects in a cluster M. Consider p to be a sensitivity function such that the closer the value is to 1, the smaller the clusters are, thus fewer objects are marked as outliers.

Determining the Approximate Distance. Our approach makes use of reservoir sampling as we described in [3]. The rationale is that D is approximated by a sample of the data objects being assessed, to identify the acceptable maximum distance between objects mapped together in a cell, in a quick and automated way. Each property has its own approximate distance D, based on the types of its instance resources. To determine the approximate distance we applied two different implementations (cf. Section 4.3.2 for their evaluation), one based on a simple sampling of triples and another one based on a modified reservoir sampler, which we call the *type-selective*. From the sample set (for both implementations), a random data object is chosen to be the *host*, and is removed from the sampler. All remaining statements in the sampler are semantically compared with the host individually, and their distance values are stored in a list. This list is then sorted in ascending order and the median value is pinpointed. We chose the median value over the mean value since the latter is influenced by outliers, whilst the former derives the central tendency, which makes it more robust.

In the first implementation, the reservoir selects a sample of triples, irrelevantly of their subject and object types. The main limitation is that, irrelevantly of the size of the reservoir, the approximate distance D value will bias towards the more frequent pairs of the subject and object types. Therefore, the sampler might not represent the broad types attached to the particular property being assessed. For instance, the *dbp:author* property has around 47% of its statements (excluding statements with literal objects) with a subject type of *dbo:Book* and an object type of *dbo:Writer*[3]. Moreover, around 24% of the triples have a subject type of *dbo:Book* and an object type *dbo:Person*. This leaves around 28% of the triples with 180 different pairs of subject and object types, which include subclasses of *dbo:Book*, *dbo:Person*, and *dbo:Writer*. Therefore, this implementation might give us many matching statements (i.e. having matching subject and objects), and thus give a low approximate D value. Our supposition was confirmed in the experiment comparing the two approaches in Section 4.3.2, where the approximate D value for the simple triple sampling was on average 0.168, whilst for the *type-selective* sampler was 0.482147. The approximate values affects the precision results, as having a low approximate D would mean that similar typed statements might not be mapped together.

Knorr et al. [8] had foreseen this problem and whilst suggesting that sampling provides a reasonable starting value for D, it cannot provide a high degree of confidence for D because of the unpredictable occurrence of outliers in the sample. In order to tackle this issue, we introduce the *type-selective* reservoir sampler. The proposed reservoir sampler is modified by adding a condition that only one statement with a certain subject type and object type can be added to the reservoir. In other words, when there are two distinct statements with matching subject types and object types, only one of these statements will be added to the reservoir. Experiments show (cf. Section 4.3.2) that the overall precision improved by around 40%, for different p fractions when using the *type-selective* reservoir sampling technique.

[3]In all of our examples we assume that each resource, subject or object, has one type.

3.1.2 Mapping Stage

The mapping stage attends to the clustering of data objects (i.e. RDF statements in our case) in cells, based on the properties described in Section 2.1. An RDF statement is chosen at random from the whole set of data objects and is placed in a random cell. This is called the *host* cell. Thereafter, every other RDF statement in the dataset is mapped to an appropriate cell by first comparing it to the data object in the host cell. We improve performance by using hash maps and sets to index:

1. the cell co-ordinates of a pair of subject and object types (identified with the variable $Map_{p,l}$ in Listing 1, where p is the index key identified by the pair and l denotes the allocated cell co-ordinates);
2. the semantic distance of two types; and
3. the non-empty cell locations.

The mapping process is described in Algorithm 1.

Semantic Similarity Measure. In order to check if an RDF statement fits in a cell with other similar RDF statements, a semantic similarity measure is used. More specifically, since we are mostly concerned about the distance between two statements, we use a normalised semantic similarity measure. The similarity (ρ) between two statements S_1 and S_2 is defined as the average of the similarity between the statements' subjects, and the similarity between the statements' objects:

$$\rho(S_1, S_2) = \left(\frac{\mathrm{sim}(S_{1\,sbj}, S_{2\,sbj}) + \mathrm{sim}(S_{1\,obj}, S_{2\,obj})}{2} \right) \quad (1)$$

This average-based definition of ρ was chosen as it represents the ideal statistical centrality of the two similarity values, i.e. the similarity between the two subject types and the similarity between the two object types.

Once the similarity of the two statements has been calculated, the normalised semantic distance [6] is calculated as follows:

$$d_{sem} = 1 - \rho(S_1, S_2) \quad (2)$$

Our approach is flexible towards the choice of the semantic similarity measure. Currently we reuse the intrinsic measures available in the semantic measures library and toolkit by Harispe et al. [4], but users can easily implement their own similarity measures. As we see in Section 4.2, different similarity measure configurations give different precision values, as the clustering is highly influenced by the initial values of the approximate D and the fraction p.

3.1.3 Colouring Stage

After mapping all data objects to the two-dimensional space, the *colouring* process colours cells to identify outlier data objects. In [8], the minimum number of objects (M) required in a cell such that data objects are not considered as outliers is calculated as:

$$M = N \cdot (1 - p) \quad (3)$$

Algorithm 1 Mapping Data Objects in a two-dimensional Space

vars: $DataObjects$, $Map_{p,l}$, $cells[][]$, $hostTriple$, $hostLocation$
$cellLength = \frac{D}{(2\sqrt{2})}$
procedure MAIN:
  for $triple$ in $DataObjects$ **do**
    $cellLocation \leftarrow$ MAPCELLS($triple$)
    $cellLocation$.counterIncrement()
    $cellLocation$.add($triple$)
function MAPCELLS($triple$)
  $cellLocation \leftarrow (-1, -1)$
  $tripleSubjectType \leftarrow$ TYPEOF($triple_{subject}$)
  $tripleObjectType \leftarrow$ TYPEOF($triple_{object}$)
  if ($Map_{p,l}$.get(<$tripleSubject, tripleObject$ >) $\neq$ null) **then**
    $cellLocation \leftarrow M_{p,l}$.get(<$tripleSubject, tripleObject$>)
  else
    $distance \leftarrow 1 -$ SIMCOMP($hostTriple$,$triple$)
    if ($distance \leq (D/2)$) **then**
      $cellLocation \leftarrow hostLocation$
    else if ($distance \leq D$) **then**
      $l_1 \leftarrow$ Encircling Cells of $hostLocation$
      $l_2 \leftarrow$ Buffer Cells of $hostLocation \setminus l_1$
      $cellLocation \leftarrow$ ALLOCATECELL(distance)
    else
      $l_1 \leftarrow$ Encircling Cells of $hostLocation$
      $l_2 \leftarrow$ Buffer Cells of $hostLocation \setminus l_1$
      $l_3 \leftarrow$ ARRAYTOSET($cells$) $\setminus l_2$
      $cellLocation \leftarrow$ ALLOCATECELL(distance)
    $Map_{p,l}$.put(<$tripleSubject, tripleObject$>, $cellLocation$)
  return $cellLocation$
function ALLOCATECELL($distance$)
  if ($distance > cellLength$)) **then**
    return a diagonal cell location (cf. Section 2.1) in l_1 or l_2
  else
    return a horizontal/vertical cell location (cf. Section 2.1) in l_1 or l_2

where N is the total number of data objects, and p is the threshold fraction value determined in the *initial* stage. The authors also define a number of conditions which we adopt in our approach and can be summarised as follows:

- if a cell has $> M$ data objects, then (a) the cell is coloured **red** and its encircling cells are coloured **pink**;
- if two adjacent cells have a total of $> M$ data objects together, but $< M + 1$ individually, then the two cells are coloured **pink**

Data objects in **red** and **pink** cells are not considered as outliers.

3.2 Time and Space Complexity Analysis

One of the main goals of this approach is to keep time and space complexity as low as possible. With regard to time complexity, we aim to achieve a *polynomial* worst-case time complexity. On the other hand, we aim that our data structures can be easily kept in memory.

3.2.1 Analysing the Data Structures used

To make sure that we keep the running time low, we make use of hash data structures (maps and sets) where the time complexity for adding and searching items is $\mathcal{O}(1)$, assuming there are no collisions. The space complexity for storing hash data structures is $\mathcal{O}(n)$. In our approach, the largest hash data structure (out of three in total) is the count of unique subject-object pairs of an assessed data property. The two-dimensional space, i.e. the cells (2D array) where data objects will be mapped, has a **worse case** space complexity of $\mathcal{O}(k^2 \times j)$, where k is the number of cells in the square array and j is the number of data objects mapped inside each cell. In practice, our algorithm is not using all the space as cells are only initialised if required when an object is mapped into a cell location for the first time. Since we are indexing (using a hash set) the occupied cells (initialised array cells with data objects), the time complexity for accessing all occupied cells in the two-dimensional space is $\mathcal{O}(n)$, where n is the size of the occupied cells hash set.

3.2.2 Analysing the Time Complexity for the Three Main Stages

After analysing the space and time complexity for the data structures used, we further analysed each of the three stages.

Initial Stage. Let R_s be the total available capacity of the reservoir sampler. The *initial* stage takes at worst $\mathcal{O}(R_s^2)$, as elements in the reservoir sampler are compared with each other. Using our *type-selective* reservoir, the time taken to calculate the approximate value D is at worst the square of the count of unique subject-object pairs of an assessed data property. The space complexity of the *type-selective* reservoir is $\mathcal{O}(R_s + p)$, where p is the size of a hash data structure indexing the subject-object pair of an added triple.

Mapping Stage. Knorr et al. state in [8], that the mapping time complexity is $\mathcal{O}(N)$, where N is the number of data objects in the data set. Our approach ensures that all mapped data objects observe the three axioms defined in Section 2.1, which means that our approach takes more than the proposed $\mathcal{O}(N)$. Mapping a *single* data object takes (**best case**) $\mathcal{O}(1)$, if all cells within the possible allocation set (i.e. either *encircling cells* or *buffer cells* in case of axiom 2, or $Layer_3$ in case of axiom 3) are free. In cases when not all cells are free, we check if the data objects in the occupied cells comply with the axioms and thus reduce the possible cell allocation. In the **worst case** this takes $\mathcal{O}(i)$, where i is a subset of all occupied cells, and each cell in i is also in the set of possible cell allocations. Overall, the time complexity of our mapping algorithm is $\mathcal{O}(N \times i)$ at worst.

Coloring Stage. The colouring stage of our approach is the same as in [8], thus the time complexity is the same as defined in the literature, i.e $\mathcal{O}(m)$.

4. EXPERIMENTS AND EVALUATIONS

Having implemented our approach, we performed three experiments focusing on complementary aspects. The first experiment that we conducted was to investigate the effect on the precision and recall when using different similarity measures. For this experiment, the algorithm was executed a number of times with different similarity configurations on a property dump (cf. Section 4.1 for an explanation on what property dumps are and how they were gen-

erated). The second experiment aims to evaluate the precision and recall in different parameter settings, i.e. different approximate D and different fraction p. Finally, we evaluate the algorithm to determine the number of automatic assessment and cleaning iterations required to achieve a steady approximate quality value, i.e. a state in which further iterations will not significantly improve the quality value.

4.1 Experiment Setup

For these experiments[4] we extracted a subset of DBpedia to generate *property dumps*. The SPARQL query in Listing 1 illustrates how property dumps are generated. The placeholder *%%PROPERTY%%* should be replaced by some concrete property (e.g. *author*). The offset is required as results are often truncated to 10k rows. A property dump is an N-Triples file with all triples related to a particular property (e.g. *dbp:author*), and the respective subject and object type for each triple. Having this property dump, all type statements were extracted to represent all statements in the dump itself. For example, the statement `<dbpedia:Franziska_Linkerhand> <dbp:author> <dbpedia:Brigitte_Reimann>` is represented as the `<dbo:Book> <dbp:author> <dbo:Writer>` pseudo triple. Although each resource might have had multiple types assigned to it, we took the least generic type that was dereferenceable. For example, the resource `<dbpedia:Brigitte_Reimann>` has 32 different types, ranging from `owl:Thing` to `yago:Winner110782791`. If on the other hand, the resource's type is unknown or cannot be dereferenced, we automatically assign `owl:Thing` as its type. These are required in our approach due to its nature of being an automated process, as the topology structure of the ontology needs to be built on the fly.

From the pseudo triples, we manually tagged which of these type pairs (by "type pair" we mean the subject and object types of an RDF statement, e.g., `dbo:Book` and `dbo:Writer`) are correct or incorrect for the corresponding property (in this case `dbp:author`). For our experiments we generated and tagged two property dumps, one for the property `dbp:author` and the other one for `dbp:publisher`. We identified 89 possibly incorrect subject-object pairs (37 had a correct object type – i.e. the object was a type or subclass type of `dbo:Writer`, 40 a correct subject type, and 12 had incorrect subject and object types) and 92 correct pairs for the `dbp:author` type. The tagging criteria was that a `dbp:author` should have a subject type of `dbo:Work` and an object type of `dbo:Person`, or any of its subclass types. With regard to the `dbp:publisher` property dump, the criterion was that for a statement to be tagged correct, the subject has to be of type `dbo:Work` and the object has to be either of type `dbo:Company` or `dbo:Organisation`, or any of its subclass types. After manually tagging the pseudo triples, we identified 99 possibly incorrect subject-object pairs (17 had a correct object type, 74 a correct subject type, and 8 had incorrect subject and object types) and 54 correct pairs. These tagged pseudo triples were then iterated and used within SPARQL `ASK` queries against the possibly incorrect statements, in order to calculate the precision and recall of our approach.

[4]All experiments and generated dumps can be found at `http://eis-bonn.github.io/Luzzu/`.

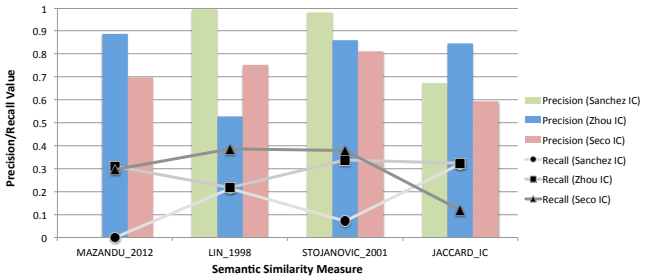

Figure 2: The precision and recall for different semantic similarity measures and configurations with a p value of 0.992.

```
PREFIX dbp:  <http://dbpedia.org/property/>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT * WHERE {
  ?instance  dbp:%%PROPERTY%%  ?author .
  ?instance  a                 ?type .
  ?type      a                 owl:Class

  FILTER NOT EXISTS {
    ?subtype  ^a               ?instance ;
              rdfs:subClassOf  ?type .
    FILTER (?subtype != ?type)
  }
  FILTER (regex(str(?type),'^http://dbpedia.org'))

  OPTIONAL {
    ?x      a  ?type2 .
    ?type2  a  owl:Class .
    FILTER NOT EXISTS {
      ?subtype2  ^a               ?x ;
                 rdfs:subClassOf  ?type2 .
      FILTER (?subtype2 != ?type2)
    }
    FILTER (regex(str(?type2),'^http://dbpedia.org'))
  }
  FILTER (isURI(?x))
}
ORDER BY ASC(?instance)
OFFSET %%OFFSET%%
```

Listing 1: Generating Property Dumps

4.2 Evaluating different Similarity Measures

The purpose of this experiment is to see how different similarity measure configurations affect the precision and the recall of detecting incorrect statements. In our approach we make use of the Semantic Measures Library & Toolkit[5] [5], and for this experiment we identified three different *intrinsic content* (IC) configurations and five different *similarity measures*. These configurations were chosen arbitrarily, as the primary aim of this experiment is not to find the best similarity technique that can be used throughout a LOD-wide assessment. On the other hand, we are planning to do so once all possible measures and other techniques (such as semantic relatedness) have been evaluated and thoroughly analysed. For this experiment we used the authors property dump (10,192 resource triples) and set the initial threshold fraction value p to 0.992. The chosen threshold fraction value corresponds to cluster groups having a minimum of 82.536 (rounded up to 83) data objects in order not to be considered as outliers. The approximate distance D was computed by the *type-selective* reservoir sampler.

[5]`http://www.semantic-measures-library.org`

Figure 2 shows the results of the precision and recall for this first experiment[6]. It can be observed that although a number of triples will be missed due to a low recall value, with regard to precision our approach performs well with different similarity measures, with the best similarity configuration giving us a precision value of almost 100%. Having a high precision and low recall would ensure a high probability of retrieving a lot of *true positive* incorrect RDF statements, i.e. marking them as outliers, from the chosen ones. The two factors affecting these measures are the approximate distance D and the threshold fraction p. If higher recall is required, then lower values of D and p have to be set.

4.3 Evaluating the Precision against different Parameters

After analysing the precision and recall for different similarity measures, we evaluate the precision and recall of our approach against different parameters. More specifically, in this experiment we aim to achieve the following:

1. prove our hypothesis that the *type-selective* reservoir sampler will give us a better precision than that of the simple triples reservoir sampler;
2. compare a manual approach of setting parameters against an automated approach (referring to the approximate D value);

All experiments in this part of the evaluation used the same similarity measure configuration, i.e. Zhou IC [16] with the Mazandu measure [9], as implemented in the Semantic Measures Library & Toolkit [5]

This experiment is split into two sub-experiments. In the first part, the two property dumps (authors and publishers) are evaluated using our approach to determine the precision and recall values. This experiment is conducted over various manually set values for the threshold fraction p and approximate distance D. The second sub-experiment evaluated the precision and recall of our proposed approach, again using the two property dumps, over a number of set values for p (as set in the previous experiment) but this time the approximate distance D is determined by the two automated approaches. For both experiments, p was set to: 0.99, 0.992, 0.994, 0.996, and 0.998, consecutively. These resulted in clusters of maximum 101.92, 81.536, 61.152, 40.768, and 20.384 data objects for the authors property dump, and 104.16, 83.328, 62.496, 41.664, and 20.832 data objects for the publishers property dump.

4.3.1 Sub-Experiment #1 – Manual Evaluation

Both dumps had different sets of D values. These sets of values where obtained as rough estimates following a manual investigation of the triple types and a manual calculation of the similarity values between the different types.

From Figure 3, we observe that on average our approach achieved around 76% precision. On the other hand, the recall values were low, with an average of 31%. Low recall was expected as the threshold fraction p was closer to 1, therefore the a cell (and its surrounding cluster) was considered a non-outlier with a low number

[6]The configuration that combined the Sanchez IC [16] and the Mazandu [9] measure was giving an infinity error related to the semantic measures library used, therefore it is not considered for these results.

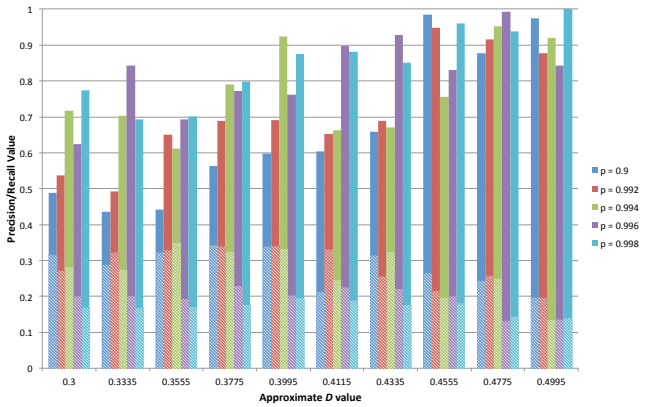

Figure 3: The precision and recall values for the authors property dump with different values for D and p. The solid bars denote precision values, whilst the striped overlapped bars denote recall.

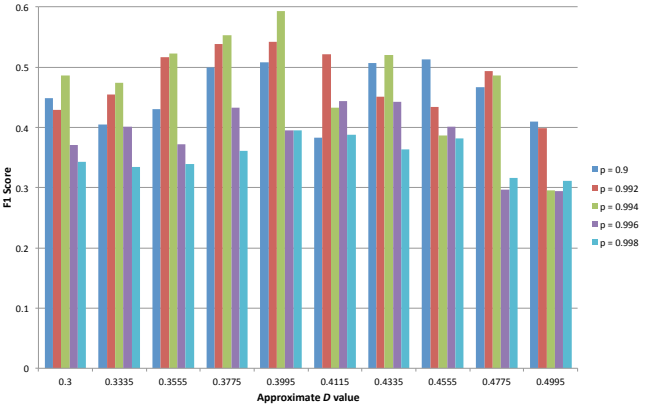

Figure 4: The F1 score authors property dump with different values for D and p.

of data objects. We also observed that increasing the approximate value D does not result in an increasing precision. For example, in Figure 3 we spot that the precision value for the D value of 0.3335 is greater than that of 0.3555 when p was set to 0.996. We investigated this further and found out that although the latter value of p (0.3555) detected 39 more outliers, the number of true positives decreased (from 189 to 182) whilst the number of false positives increased by 42 data objects. This slight change in true positives and false positives was expected as the data objects will try to cluster with those whose distance is the smallest. Therefore, the change in D might have moved some objects from one cell to another with the consequence that a previously non-outlier cell (or two adjacent) is now marked as an outlier, as a number of data objects might have moved to other cells. Another important factor that can affect the precision and recall values is the choice of the host object. Figure 4 represents the F1 score for the authors property dump manual experiment, showing an average of almost 0.43 for this harmonic mean score.

A similar trend of a higher precision than recall was also noted in Figure 5, where an acceptable average of 70% precision was

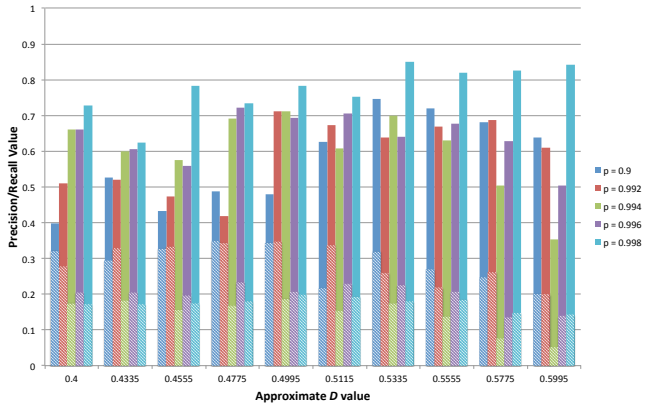

Figure 5: The precision and recall values for the publishers property dump with different values for D and p. The solid bars denote precision values, whilst the striped overlapped bars denote recall.

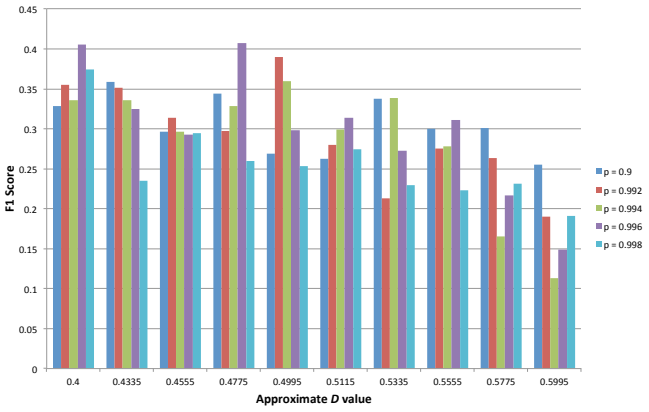

Figure 6: The F1 score publishers property dump with different values for D and p.

achieved, whilst recall values were still considerably below the 50% mark for the triples with a `dbp:publisher` property. We observed that the approximate D value with 0.4995 and a threshold p of 0.992 gave the best precision and recall. Similar to the previous experiment, these values are only applicable to the property being assessed, as a similarity distance has to be calculated between the subject and object types being used with the assessed property. The F1 score in this experiment is around 0.3 (cf. Figure 6). Overall, these results show that whilst our algorithm is retrieving many relevant data objects (i.e. a high number of *true positives*), there exist more possibly incorrect triples in a dataset that need to be detected.

4.3.2 Sub-Experiment #2 – Automatic Evaluation

This sub-experiment contributes towards our hypothesis that the *type-selective* reservoir sampler will give us a better precision than the simple triples reservoir sampler. From Figure 7 and Figure 8 we observe that the *type-selective* sampler outperforms its simpler counterpart for all p values. On the other hand, we observe that the simple reservoir sampler always gives a slightly better recall percentage than the *type-selective* one. This is due to the low approximate D values identified by the simple reservoir sampler. Low

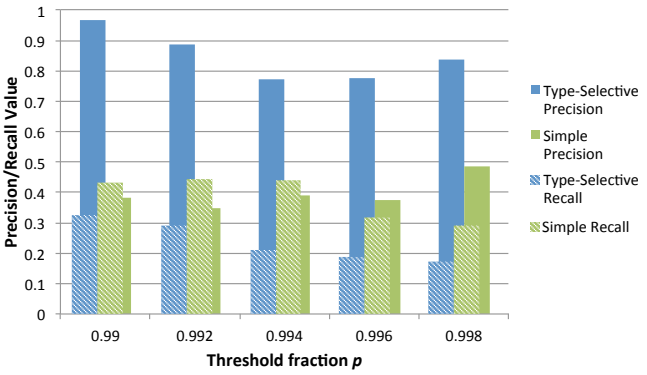

Figure 7: The precision and recall values for the authors property dump with different values for p and a generated D value.

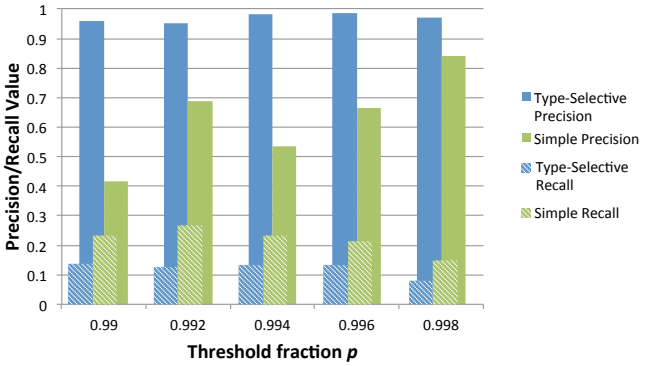

Figure 8: The precision and recall values for the publishers property dump with different values for p and a generated D value.

approximate D values mean that less data objects get mapped together in cells.

Overall, in Figure 9 and Figure 10 we compare the manual approach of defining an approximate D value against that generated by the *type-selective* reservoir. Results show that in most cases the automatic *type-selective* reservoir sampler gives a better precision, whilst the manual approach gives a better recall. This is because the slightest decimal fraction in the approximate D value can make a difference to the mapping of data objects to cells.

4.4 Approximating Quality for Incorrect RDF Statements

The rationale of this experiment is to evaluate and see the approximate number of iterations required until the algorithm produces no more true positive outliers. We automated the process that increases and decreases the threshold fraction value p after each iteration. This process averages the count of the pair of types, which we said that it would be the number of minimum objects in a cluster and then used $(M - N)/N = p$ to find the fraction value p. Using the tagged pseudo triples we simulated the *real quality value* and the calculated an *approximate quality value*. The former is calculated as:

$$qv_{real} = 1 - \left(\frac{\#truePositives}{\#totalTriples}\right) \quad (4)$$

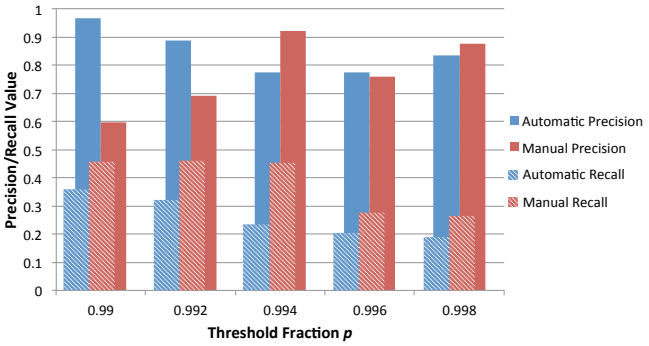

Figure 9: Precision and recall values for the authors property dump comparing the manual results (with a D value of 0.3995) against the automatic results for multiple values of the fraction p.

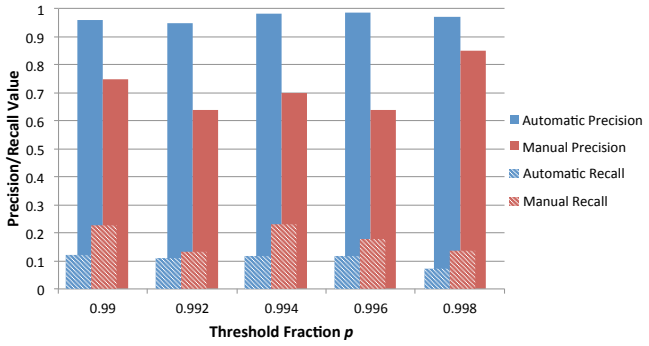

Figure 10: Precision and recall values for the publishers property dump comparing the manual results (with a D value of 0.5335) against the automatic results for multiple values of the fraction p.

whilst the latter is calculated as:

$$qv_{approximate} = 1 - \left(\frac{\#allOutliers}{\#totalTriples}\right) \quad (5)$$

Figure 11 and Figure 12 depict two graphs with iterations and their corresponding quality values. Both graphs show that the algorithm needs between 5 and 6 iterations until a steady approximate quality value can be achieved. Even though the ideal value 1 was not achieved, we believe that the reported approximate values are sufficient, as *false positives* are unavoidable when using unsupervised clustering techniques.

5. RELATED WORK

Various research efforts have tackled the problem of detecting incorrect RDF statements using different techniques. These include *statistical distribution* [10], *schema enrichment* [15, 11] and *crowdsourcing* [13, 1]. Outlier detection techniques such as [14] are used to validate the correctness of data literals in RDF statements, which is out of the scope of this research as our approach considers only statements where the subject and object are resources.

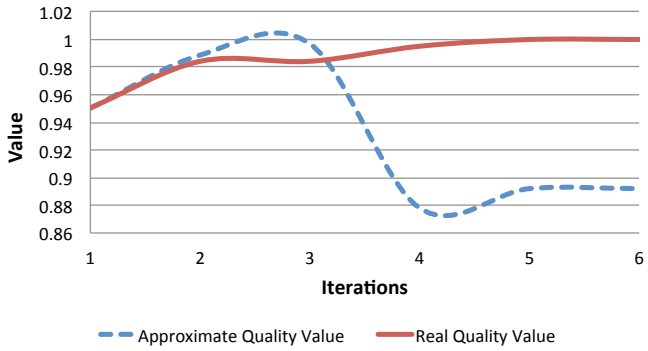

Figure 11: Iterations and Values (Approximate and Real) for the Quality Assessment of Incorrect RDF Triples – Authors.

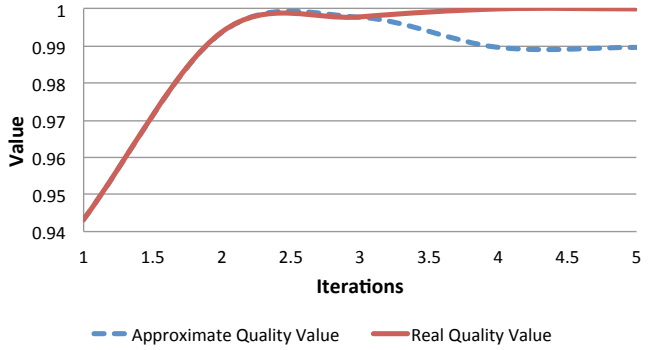

Figure 12: Iterations and Values (Approximate and Real) for the Quality Assessment of Incorrect RDF Triples – Publishers.

Statistical Distribution. Paulheim et al. provide in [10] an algorithm based on the statistical distribution of types over properties in order to identify possibly faulty statements, whose subject and object are resources and not literals. The distributions were deducted in order to predict the probability of the types of the subject and object of a property, and then used to determine, with some confidence, the correctness value of a triple statement. Their three step approach first computes the frequency of the predicate and object combination in order to identify those statements that have a low value. Cosine similarity is then used to calculate a confidence score based on the statement's subject type probability and the object type probability. Finally, a threshold value is applied to mark those statements that are potentially incorrect. The authors report a precision of above 90% (0.9), which is comparable to the precision of our approach when using the *type-selective* reservoir sampler. Whilst their approach is said to identify and remove around 13,000 incorrect triples from DBpedia, we cannot compare how their approach fares with regard to recall per property. This means that we cannot quantify the number of false negatives produced by this approach. Our approach uses semantic similarity to identify whether a statement could be a possibly incorrect statement or not, instead of statistical distribution probabilities. Therefore, our similarity approach takes into consideration the semantic topology of types and not just their statistical usage.

Schema enrichment. Schema enrichment is also a popular technique to detect incorrect statements. Töpper et al. [11] first automatically enrich a knowledge base schema (their use case being DBpedia) with additional axioms (including domain-range restrictions and class disjointness) before detecting incorrect RDF statements in the knowledge base itself. Such an approach requires external knowledge (in this case Wikipedia) in order to enrich the ontology, which is not required in our case as long as the underlying schema of the knowledge base being assessed is dereferenceable. Similarly, Zaveri et al. [15] apply a semi-automated schema enrichment technique before detecting incorrect triples.

Crowdsourcing. *WhoKnows?* [13] is a crowdsourcing game where users, possibly unknowingly, contribute towards identifying inconsistent, incorrect and doubtful facts in DBpedia. Such crowdsourcing efforts ensure that the quality of a dataset can be improved with more accuracy, as a human assessor can identify such problems even from a subjective point of view. During the evaluation, the users identified 342 triples that were potentially inconsistent from a set of overall 4,051 triples. These were distributed as follows: 77 identified incorrect triples were generated due to bugs in the code, 144 were wrongly identified as incorrect triples (false positives) and 121 were verified as true wrong triples (true positives). Based on these values, the precision (without the 77 incorrect triples) is around 0.46. A similar crowdsourcing effort was undertaken by Acosta et al. in [1]. They used pay-per-hit micro tasks as a means of improving the outcome of crowdsourcing efforts. Their evaluation focuses on checking the correctness of the object values and their data types, and the correctness of interlinking with related external sources, thus making it incomparable to our approach. On the other hand we learned two things about crowdsourcing: (1) it requires a mix of expert and non-expert users, and (2) it is time-consuming and costly. In contrast to crowdsourcing, our approach provides a good precision while at the same being time efficient (e.g. both property dumps used in the evaluation had over 10k triples and it takes around $\pm$ 3 minutes to compute outliers).

6. FINAL REMARKS AND FUTURE WORK

Improving the quality of linked datasets has a significant impact on the Web of Data and its users, including producers as well as consumers. Having good quality datasets ensures their reusability and thus helps in decreasing the number of duplicate and redundant resources on the Web. The semantic lifting process can produce a number of incorrect and inconsistent triples that affect the quality and thus reusability of the Web of Data resources.

In this article we presented a time and space efficient approach that detects potentially incorrect RDF statements in a dataset using a distance-based clustering technique to identify outliers. The main contribution of our approach is that we improved and automated the technique from [8] whilst applying it to a Linked Data scenario. The process of finding potentially incorrect statements for a certain propriety includes:

- the *initial* stage, where a reservoir sampling technique is used to determine a base approximate distance;
- the *mapping* stage, where RDF statements are mapped to cells in a two-dimensional grid based on some heuristic properties;
- the *colouring* stage, where the final clustering and outlier identification is done.

We also provided an empirical evaluation of complementary aspects of the three mentioned stages. First we looked into how different similarity measures affect the precision and recall values. We observe that, in most cases, the similarity measures maintain a precision of more than 0.7 and a recall of around 0.4, which makes the approach very suitable to assist knowledge engineers in quality improvement. In the second experiment we evaluated a manual approach against two automated approaches for defining the approximate clustering distance D. We observed that the proposed *type-selective* reservoir sampler outperforms the other approaches in most cases. Finally, we automated the scenario where a user assesses a property data dump for its quality w.r.t. correct usage of domains and ranges, cleans the dataset based on the results returned by the algorithm, and iterates until our approach returns no more outliers. The results were encouraging as the approximate (automated) value was not far off the simulated actual value, whilst the number of required assessment-cleaning iterations was kept as low as 6. We provided our implementation as a metric within the *Luzzu*[7] quality assessment framework for Linked Data [2]. The rationale of Luzzu is to provide an integrated platform that:

- assesses Linked Data quality using a library of generic and user-provided domain specific quality metrics in a scalable manner;
- provides queryable quality metadata on the assessed datasets;
- assembles detailed quality reports on the assessed datasets.

While providing satisfactory results, our approach has a number of limitations that we are currently addressing. The presented approach is based on semantic similarity measures and, to the best of our knowledge, there is currently no technique that allows us to compare two concepts from two different schemas. Also, using semantic similarity measures we are limited to '*is-a*' relations. In future work, we aim to experiment with this approach using semantic relatedness measures instead, thus our distance based measure will also consider the semantic relationships between two terms, such as `owl:equivalentClass`.

One major stumbling block we encountered during our evaluation was the lack of evidence (i.e. available implementation or comparison results like precision and recall) provided by related work which would enable us to compare it to our approach. Ideally, benchmarks and gold standards should be created that allow researchers to evaluate their tool/approach against it, similarly to what is done in the fields of information retrieval or question answering.

Acknowledgments

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

[7]Sources: `https://github.com/EIS-Bonn/Luzzu`; Website: `http://eis-bonn.github.io/Luzzu/`

References

1. Acosta, M. et al. Crowdsourcing Linked Data Quality Assessment. In: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*. Ed. by H. Alani et al. Vol. 8219. Lecture Notes in Computer Science. Springer, 2013, pp. 260–276. http://dx.doi.org/10.1007/978-3-642-41338-4_17.
2. Debattista, J., Lange, C., Auer, S. LUZZU – A Framework for Linked Data Quality Assessment. In: (2014). http://arxiv.org/abs/1412.3750.
3. Debattista, J. et al. Quality Assessment of Linked Datasets Using Probabilistic Approximation. In: *12th European Semantic Web Conference Proceedings*. 2015.
4. Harispe, S. et al. The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies. In: Bioinformatics 30(5) (2014), pp. 740–742. eprint: http://bioinformatics.oxfordjournals.org/content/30/5/740.full.pdf+html. http://bioinformatics.oxfordjournals.org/content/30/5/740.abstract.
5. Harispe, S. et al. A Framework for Unifying Ontology-based Semantic Similarity Measures: a Study in the Biomedical Domain. In: Journal of Biomedical Informatics In press (2013).
6. Harispe, S. et al. Semantic Measures for the Comparison of Units of Language, Concepts or Entities from Text and Knowledge Base Analysis. In: ArXiv 1310.1285 (Oct. 2013). arXiv: 1310.1285. http://arxiv-web3.library.cornell.edu/abs/1310.1285.
7. Hodge, V., Austin, J. A Survey of Outlier Detection Methodologies. In: Artif. Intell. Rev. 22(2) (Oct. 2004), pp. 85–126. http://dx.doi.org/10.1023/B:AIRE.0000045502.10941.a9.
8. Knorr, E. M., Ng, R. T., Tucakov, V. Distance-based Outliers: Algorithms and Applications. In: The VLDB Journal 8(3-4) (Feb. 2000), pp. 237–253. http://dx.doi.org/10.1007/s007780050006.
9. Mazandu, G. K., Mulder, N. J. A Topology-Based Metric for Measuring Term Similarity in the Gene Ontology. In: Advances in Bioinformatics 2012 (2012), pp. 1–17. http://www.hindawi.com/journals/abi/2012/975783/.
10. Paulheim, H., Bizer, C. Improving the Quality of Linked Data Using Statistical Distributions. In: Int. J. Semant. Web Inf. Syst. 10(2) (Apr. 2014), pp. 63–86. http://dx.doi.org/10.4018/ijswis.2014040104.
11. Töpper, G., Knuth, M., Sack, H. DBpedia Ontology Enrichment for Inconsistency Detection. In: *Proceedings of the 8th International Conference on Semantic Systems*. I-SEMANTICS '12. Graz, Austria: ACM, 2012, pp. 33–40. http://doi.acm.org/10.1145/2362499.2362505.
12. Vitter, J. S. Random Sampling with a Reservoir. In: ACM Trans. Math. Softw. (1985).
13. Waitelonis, J. et al. WhoKnows? - Evaluating Linked Data Heuristics with a Quiz that Cleans Up DBpedia. In: International Journal of Interactive Technology and Smart Education (ITSE) 8(3) (2011), pp. 236–248.
14. Wienand, D., Paulheim, H. Detecting Incorrect Numerical Data in DBpedia. In: *11th ESWC 2014 (ESWC2014)*. 2014. http://data.semanticweb.org/conference/eswc/2014/paper/research/39.
15. Zaveri, A. et al. User-driven Quality Evaluation of DBpedia. In: *Proceedings of the 9th International Conference on Semantic Systems*. I-SEMANTICS '13. Graz, Austria: ACM, 2013, pp. 97–104. http://doi.acm.org/10.1145/2506182.2506195.
16. Zhou, Z., Wang, Y., Gu, J. A New Model of Information Content for Semantic Similarity in WordNet. In: *FGCNS'08 Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking Symposia - Volume 03*. IEEE Computer Society, Dec. 2008, pp. 85–89.