Undefined 0 (2015) 1–0
IOS Press

Detecting Linked Data Quality Issues via Crowdsourcing: A DBpedia Study

Maribel Acosta [a], Amrapali Zaveri [b] Elena Simperl [c], Dimitris Kontokostas [b], Fabian Flöck [d], Jens Lehmann [b]

[a] *Institute AIFB, Karlsruhe Institute of Technology, Germany*
E-mail: maribel.acosta@kit.edu
[b] *Institut für Informatik, AKSW, Universität Leipzig, Germany*
E-mail: {zaveri,kontokostas,lehmann}@informatik.uni-leipzig.de
[c] *Web Science and Internet Research Group, University of Southampton, United Kingdom*
E-mail: e.simperl@soton.ac.uk
[d] *Computational Social Science Group, GESIS - Leibniz Institute for the Social Sciences, Germany*
E-mail: fabian.floeck@gesis.org

Abstract. In this paper we examine the use of crowdsourcing as a means to master Linked Data quality problems that are difficult to solve automatically. We base our approach on the analysis of the most common errors encountered in Linked Data sources, and a classification of these errors according to the extent to which they are likely to be amenable to crowdsourcing. We then propose and compare different crowdsourcing approaches to identify these Linked Data quality issues, employing the DBpedia dataset as our use case: (i) a contest targeting the Linked Data expert community, and (ii) paid microtasks published on Amazon Mechanical Turk. We secondly focus on adapting the *Find-Fix-Verify* crowdsourcing pattern to exploit the strengths of experts and lay workers. By testing two distinct *Find-Verify* workflows (lay users only and experts verified by lay users) we reveal how to best combine different crowds' complementary aptitudes in quality issue detection. The results show that a combination of the two styles of crowdsourcing is likely to achieve more efficient results than each of them used in isolation, and that human computation is a promising and affordable way to enhance the quality of Linked Data.

1. Introduction

Many would consider Linked Data (LD) to be one of the most important technological trends in data management of the last decade [16]. However, seamless consumption of LD in applications is still very limited given the varying quality of the data published in the Linked Open Data (LOD) Cloud [18,44]. This is the result of a combination of data- and process-related factors. The data sets being released into the LOD Cloud are – apart from any factual flaws they may contain – very diverse in terms of formats, structure, and vocabulary. This heterogeneity and the fact that some kinds of data tend to be more challenging to lift to RDF than others make it hard to avoid errors, especially when the translation happens automatically. Simple issues like syntax errors or duplicates can be easily identified and repaired in a fully automatic fashion. However, data quality issues in LD are more challenging to detect. Current approaches to tackle these problems still require expert human intervention, e.g., for specifying rules [14] or test cases [21], or fail due to the context-specific nature of quality assessment, which does not lend itself well to general workflows and rules that could be executed by a computer program. In this paper, we explore an alternative data curation strategy, which is based on crowdsourcing.

Crowdsourcing [19] refers to the process of solving a problem formulated as a task by reaching out to a large network of (often previously unknown) people. One of the most popular forms of crowdsourcing are 'microtasks' (or 'microwork'), which consists on dividing a task into several smaller subtasks that can be independently solved. Conditional on the tackled problem, the level of task granularity can vary (microtasks whose results need to be aggregated vs. macrotasks, which require filtering to identify the most valuable contributions); as can the incentive structure (e.g., payments per unit of useful work vs. prizes for top participants in a contest). Another major design decision in the crowdsourcing workflow is the selection of the crowd. While many (micro)tasks can be performed by untrained workers, others might require more skilled human participants, especially in specialized fields of expertise, such as LD. Of course, expert intervention usually comes at a higher price; either in monetary rewards or in the form of effort to recruit participants in another setting, such as volunteer work. Microtask crowdsourcing platforms such as Amazon Mechanical Turk (MTurk)[1] on the other hand offer a formidable and readily-available workforce at relatively low fees.

In this work, we crowdsource three specific LD quality issues. We did so building on previous work of ours [43] which analyzed common quality problems encountered in Linked Data sources and classified them according to the extent to which they could be amenable to crowdsourcing. The first research question explored is hence: **RQ1:** *Is it feasible to detect quality issues in LD sets via crowdsourcing mechanisms?* This question aims at establishing a general understanding if crowdsourcing approaches can be used to find issues in LD sets and if so, to what degree they are an efficient and effective solution. Secondly, given the option of different crowds, we formulate **RQ2:** *In a crowdsourcing approach, can we employ unskilled lay users to identify quality issues in RDF triple data or to what extent is expert validation needed and desirable?* As a subquestion to RQ2, we also examined which type of crowd is most suitable to detect which type of quality issue (and, conversely, which errors they are prone to make). With these questions, we are interested (i) in learning to what extent we can exploit the cost-efficiency of lay users, or if the quality of error detection is prohibitively low. We (ii) investigate how well experts generally perform in a crowdsourcing setting and if and how they outperform lay users. And lastly, (iii) it is of interest if one of the two distinct approaches performs well in areas that might not be a strength of the other method and crowd.

[1]`https://www.mturk.com/`

To answer these questions, we (i) first launched a **contest** that acquired 58 experts knowledgeable in Linked Data to find and classify erroneous RDF triples from DBpedia (Section 4.1). They inspected $68,976$ triples in total. These triples were then (ii) submitted as **paid microtasks** on MTurk to be examined by workers on the MTurk platform in a similar way (Section 4.2). Each approach (contest and paid microtasks) makes several assumptions about the audiences they address (the 'crowd') and their skills. This is reflected in the design of the crowdsourcing tasks and the related incentive mechanisms. The results of both crowds were then compared to a manually created gold standard.

The results of the comparison of experts and turkers, as discussed in Section 5, indicate that (i) untrained crowdworkers are in fact able to spot certain quality issues with satisfactory precision; that (ii) experts perform well locating two but not the third type of quality issues given, and that lastly (iii) the two approaches reveal complementary strengths.

Given these insights, **RQ3** was formulated: *How can we design better crowdsourcing workflows using lay users or experts for curating LD sets, beyond one-step solutions for pointing out quality flaws?* To do so, we adapted the crowdsourcing pattern known as *Find-Fix-Verify*, which has been originally proposed by Bernstein et al. in [3]. Specifically, we wanted to know: can (i) we enhance the results of the LD quality issue detection through lay users by adding a subsequent step of cross-checking (*Verify*) to the initial *Find* stage? Or is it (ii) even more promising to combine experts and lay workers by letting the latter *Verify* the results of the experts' *Find* step, hence drawing on the crowds' complementary skills for deficiency identification we recognized before?

Accordingly, the results of both *Find* stages (expert and workers) – in the form of sets of triples identified as incorrect, marked with the respective errors – were fed into a subsequent *Verify* step, carried out by MTurk workers (Section 4.3). The task consisted solely of the rating of a formerly indicated quality issue for a triple as correctly or wrongly assigned. This *Verify* step was, in fact, able to improve the precision of both *Find* stages substantially. In particular, the experts' *Find* stage results could be improved to precision levels of around 0.9 in the *Verify* stage for two error types which showed to score much lower for an expert-only *Find* approach. The worker-worker *Find-Verify* strategy yielded also better results than the *Find*-only worker approach, and for one error type even reached slightly better precision than the expert-worker model. All in all, we show that (i) a *Find-Verify* combination of experts and lay users is likely to produce the best results, but that (ii) they are not superior to expert-only evaluation in all cases. We demonstrate also that (iii) lay users-only *Find-Verify* approaches can be a viable alternative for detection of LD quality issues if experts are not available and that they certainly outperform *Find*-only lay user workflows.

Note that we did not implement a *Fix* step in this work, as correcting the greatest part of the found errors via crowdsourcing is not the most cost-efficient method of addressing these issues. Thus, we argue in Section 4, a majority of errors can and should be addressed already at the level of individual wrappers leveraging datasets to LD.

To understand the strengths and limitations of crowdsourcing in this scenario, we further executed automated baseline approaches to compare them to the results of our crowdsourcing experiments. We show that while they may be amenable to pre-filtering RDF triple data for ontological inconsistencies (thus potentially decreasing the amount of cases necessary to be browsed in the *Find* stage), a substantial part of quality issues can only be addressed via human intervention.

Contributions

This paper is an extension to previous work of ours [1], in which we presented the results of combining LD experts and lay users from MTurk when detecting quality issues in DBpedia. The novel contributions of our current work can be summarized as follows:

– Definition of the problem of classifying RDF triples into quality issues.
– Formalization of the proposed approach: The adaptation of the *Find-Fix-Verify* pattern is formalized for the problem of detecting quality issues in RDF triples.
– Introduction of a new crowdsourcing workflow that solely relies on microtask crowdsourcing to detect LD quality issues.
– Analysis of the properties of our approaches to generate microtasks for triple-based quality assessment.
– Empirical evaluation of the proposed workflow.
– Inclusion of a new baseline study by executing the state-of-the-art solution RDFUnit [21], a test-based approach to detect LD quality issues either manually or (semi-)automatically.

Structure of the paper

In Section 2, we discuss the type of LD quality issues that are studied in this work. Section 3 briefly introduces the crowdsourcing methods and related concepts that are used throughout the paper. Our approach is presented in Section 4, and is empirically evaluated in Section 5. In Section 6 we summarize the findings of our experimental study and provide answers to the formulated research questions. Related work is discussed in Section 7. Conclusions and future work are presented in Section 8.

2. Linked Data Quality Issues

The Web of Data spans a network of data sources of varying quality. There are a large number of high-quality data sets, for instance, in the life-science domain, which are the result of decades of thorough curation and have been recently made available as Linked Open Data[2]. Other data sets, however, have been (semi-)automatically translated into RDF from their primary sources, or via crowdsourcing in a decentralized process involving a large number of contributors, for example DBpedia [23]. While the combination of machine-driven extraction and crowdsourcing was a reasonable approach to produce a baseline version of a greatly useful resource, it was also the cause of a wide range of quality problems, in particular in the mappings between Wikipedia attributes and their corresponding DBpedia properties.

Our analysis of Linked Data quality issues focuses on DBpedia as a representative data set for the broader Web of Data due to the diversity of the types of errors exhibited and the vast domain and scope of the data set. In our previous work [44], we compiled a list of data quality dimensions (criteria) applicable to Linked Data quality assessment. Afterwards, we mapped these dimensions to DBpedia [43]. A sub-set of four dimensions of the original framework were found particularly relevant in this setting: *Accuracy*, *Relevancy*, *Representational-Consistency* and *Interlinking*. To provide a comprehensive analysis of DBpedia quality, we further divided these four categories of problems into sub-categories. For the purpose of this paper, from these categories we chose the following three triple-level quality issues.

[2]`http://beta.bio2rdf.org/`

Object incorrectly/incompletely extracted. Consider the triple: (dbpedia:Rodrigo_Salinas, dbpedia-owl:birthPlace, dbpedia:Puebla_F.C.). The DBpedia resource is about the person 'Rodrigo Salinas', with the incorrect value of the birth place. Instead of extracting the name of the city or country from Wikipedia, the stadium name Puebla F.C, is extracted.

Datatype or language tag incorrectly extracted. This category refers to triples with an incorrect datatype for a typed literal. For example, consider the triple: (dbpedia:Oreye, dbpedia-owl:postalCode, "4360"@en). The datatype of the literal "4360" is incorrectly identified as english instead of integer.

Incorrect link. This category refers to RDF triples whose association between the subject and the object is incorrect. Erroneous interlinks can associate values within a dataset or between several data sources. This category of quality issues also includes links to external Web sites or other external data sources such as Wikimedia, Freebase, GeoSpecies or links generated via the Flickr wrapper are incorrect; that is, they do not show any related content pertaining to the resource.

These categories of quality problems occur pervasively in DBpedia. These problems might be present in other data sets which are extracted in a similar fashion as DBpedia. Given the diversity of the situations in which they can be instantiated (broad range of datatypes and object values) and their sometimes deeply contextual character (interlinking), assessing them automatically is challenging. In the following we explain how crowdsourcing could support quality assessment processes.

3. Crowdsourcing Preliminaries

3.1. Types of Crowdsourcing

The term crowdsourcing was first proposed by Howe [19] that consists on a problem-solving mechanism in which a task is performed by an *"an undefined (and generally large) network of people in the form of an open call."* Nowadays, many different forms of crowdsourcing have emerged, e.g., microtask, contests, macrotask, crowdfunding, among others; each form of crowdsourcing is designed to target particular types of problems and reaching out to different crowds. In the following we briefly describe contest-based and microtask crowdsourcing, the two crowdsourcing methods studied in this work.

3.1.1. Contest-based Crowdsourcing

A contest reaches out to a crowd to solve a given problem and rewards the best ideas. It exploits competition and intellectual challenge as main drivers for participation. The idea, originating from open innovation, has been employed in many domains, from creative industries to sciences, for tasks of varying complexity (from designing logos to building sophisticated algorithms). In particular, contests as means to successfully involve experts in advancing science have a long-standing tradition in research, e.g., the Darpa challenges[3] and NetFlix.[4] Usually, contests as crowdsourcing mechanisms are open for a medium to long period of time in order to attract high quality contributions. Contests may apply different reward models, but a common modality is to define one main prize for the contest winner.

We applied this contest-based model to mobilize an expert crowd consisting of researchers and Linked Data enthusiasts to discover and classify quality issues in DBpedia. The reward mechanism applied in this contest was "one-participant gets it all". The winner was the participant who covered the highest number of DBpedia resources.

3.1.2. Microtask Crowdsourcing

This form of crowdsourcing is applied to problems which can be broken down into smaller units of work (called 'microtasks'). Microtask crowdsourcing works best for tasks that rely primarily on basic human abilities, such as visual and audio cognition or natural language understanding, and less on acquired skills (such as subject-matter knowledge).

To be more efficient than traditional outsourcing (or even in-house resources), microtasks need to be highly parallelized. This means that the actual work is executed by a high number of contributors in a decentralized fashion;[5] this not only leads to significant improvements in terms of time of delivery, but also offers a means to cross-check the accuracy of the answers (as each task is typically assigned to more than one person). Collecting answers from different workers allow for techniques such as majority voting (or other aggregation methods) to automatically identify accurate responses. The most common reward model in micro-

[3] `http://www.darpa.mil/About/History/Archives.aspx`

[4] `http://www.netflixprize.com/`

[5] More complex workflows, though theoretically feasible, require additional functionality to handle task dependencies.

task crowdsourcing implies small monetary payments for each worker who has successfully solved a task.

In our work, we used microtask crowdsourcing as a fast and cost-efficient way to examine the three types of DBPedia errors described in Section 2. We provided specific instructions to workers about how to assess RDF triples according to the three previous quality issues. We reached out to the crowd of the microtask marketplace Amazon Mechanical Turk (MTurk). In the following we present a summary of the relevant MTurk terminology:

– *Requester:* Submits tasks to the platform (MTurk).
– *Human Intelligence Task (HIT):* Work unit in MTurk and refer to a single microtask. A HIT is a self-contained task submitted by a requester.
– *Worker:* Human provider who solves HITs.
– *Assignments:* Number of different workers to be assigned to solve each HIT. This allows to collect multiple answers for each question. A worker can solve a HIT only once.
– *Question:* A HIT can be composed of several questions. In the remainder of this paper, we refer to *task granularity* as the number of questions contained within a HIT.
– *Payment:* Monetary reward granted to a worker for successfully completing a HIT. Payments are defined by the requester, taking into consideration the complexity of the HIT, mainly defined as the time that workers have to spend to solve the task.
– *Qualification type or worker qualification:* Requesters may specify parameters to prohibit certain workers to solve tasks. MTutk provide a fixed set of qualification types, including "Approval Rate" defined as the percentage of tasks successfully solved by a worker. In addition, requesters can create customized qualification types.

3.2. Crowdsourcing Pattern Find-Fix-Verify

The *Find-Fix-Verify* pattern [3] consists on dividing a complex human task into a series of simpler tasks that are carried out in a three-stage process. Each stage in the *Find-Fix-Verify* pattern corresponds to a verification step over the outcome produced in the immediate previous stage. The first stage of this crowdsourcing pattern, *Find*, asks the crowd to identify portions of data that require attention depending on the task to be solved. In the second stage, *Fix*, the crowd corrects the elements belonging to the outcome of the previous stage. The *Verify* stage corresponds to a final quality control iteration.

Originally, this crowdsourcing pattern was introduced in Soylent [3], a human-enabled word processing interface that contacts microtask workers to edit and improve parts of a document. The tasks studied in Soylent include: text shortening, grammar check, and unifying citation formatting. For example, in the Soylent text shortening task, microtasks workers in *Find* stage are asked to to identify portions of text that can potentially be reduced in each paragraph. Candidate portions that meet certain consensus degree among workers move on to the next step. In the *Fix* stage, workers must shorten the previously identified portions of paragraphs. All the rewrites generated are assessed by workers to select the most appropriate one without changing the meaning of the original text.

The *Find-Fix-Verify* pattern has proven to produce reliable results since each stage exploits independent agreement to filter out potential low-quality answers from the crowd. In addition, this approach is efficient in terms of the number of questions asked to the paid microtask crowd, therefore the costs remain competitive with other crowdsourcing alternatives.

In scenarios in which crowdsourcing is applied to validate the results of machine computation tasks, question filtering relies on specific thresholds or historical information about the likelihood that human input will significantly improve the results generated algorithmically. *Find-Fix-Verify* addresses tasks that initially can be very complex (or very large), like in our case the discovery and classification of various types of errors in DBpedia.

The *Find-Fix-Very* pattern is highly flexible, since each stage can employ different types of crowds, as they require different skills and expertise [3].

4. Our Approach: Crowdsourcing Linked Data Quality Assessment

Our work on human-driven Linked Data quality assessment focuses on applying crowdsourcing techniques to annotate RDF triples with their corresponding quality issue. Given a set of quality issues $\mathcal{Q}$ and a set $\mathcal{T}$ of RDF triples to be assessed, we formally define the annotation of triples with their corresponding quality issues as follows.

Definition 1. (Problem Definition: Mapping RDF Triples to Quality Issues). *Given $\mathcal{T}$ a set of RDF triples and $\mathcal{Q}$ a set of quality issues, a mapping of triples to quality issues is defined as a partial function*

$\phi : \mathcal{T} \rightarrow 2^{\mathcal{Q}}$. *$\phi(t)$ denotes the quality issues associated with $t \in \mathcal{T}$. In particular, when $\phi(t) \neq \emptyset$ the triple t is considered 'incorrect', otherwise it can be affirmed that t is 'correct'.*

In order to provide an efficient crowdsourcing solution to the problem presented in Definition 1, we applied a variation of the crowdsourcing pattern *Find-Fix-Verify* [3]. As discussed in Section 3, this crowdsourcing pattern allows for increasing the overall quality of the results while maintaining competitive monetary costs when applying other crowdsourcing approaches. Our implementation of the *Find-Fix-Verify* pattern is tailored to assess the quality of Linked Data sets that are automatically created from other sources. Such is the case of DBpedia [24], a data set created by extracting knowledge from Wikipedia via declarative mediator/wrapper pattern. The wrappers are the result of a crowdsourced community effort of contributors to the DBpedia project. To crowdsource the assessment of triples of data sets like DBpedia, we devise a two-fold approach including the following stages: *Find* and *Verify*. In the *Find* stage, the crowd was requested to detect LD quality issues in a set of RDF triples, and annotate them with the corresponding issue(s) if applicable. We define the Find Stage as follows:

Definition 2. (Find Stage). *Given a set $\mathcal{T}$ of RDF triples and a set $\mathcal{Q}$ of quality issues, the Find stage consists on crowdsourcing the mappings $\dot{\phi} : \mathcal{T} \rightarrow 2^{\mathcal{Q}}$. The input of the Find stage is represented as $\mathcal{F}_i = (\mathcal{T}, \mathcal{Q})$, and the output $\mathcal{F}_o = (\mathcal{T}, \dot{\phi}(\mathcal{T}))$.*

The outcome of this stage – triples judged as 'incorrect' – is then assessed in the *Verify* stage, in which the crowd confirms/denies the presence of quality issues in each RDF triple processed in the previous stage. We define the Verify Stage as follows:

Definition 3. (Verify Stage). *Given a set $\mathcal{T}$ of RDF triples and mappings $\dot{\phi}(T)$, the Verify stage consists on crowdsourcing mappings as follows $\ddot{\phi} : \dot{\phi}(\mathcal{T}) \rightarrow \dot{\phi}(\mathcal{T})$. The input of the Verify stage is represented as, $\mathcal{V}_i = (\mathcal{T}, \dot{\phi}(\mathcal{T}))$ which corresponds to the output of the Find stage ($\mathcal{V}_i = \mathcal{F}_o$), and the output of the Verify stage is represented as $\mathcal{V}_o = (\mathcal{T}, \ddot{\phi}(\mathcal{T}))$.*

The *Fix* stage originally proposed in the *Find-Fix-Verify* pattern is out of the scope of this paper, since the main goal of this work is identifying quality issues. In addition, since our work is designed for data sets extracted automatically via wrappers, it is highly probable that the quality issues detected for a certain triple might also occur in the set of triples that were generated via the same wrapper. Therefore, a more efficient solution to implement the *Fix* stage could consist of adjusting the wrappers that caused the issue in the first place, instead of crowdsourcing the correction of each triple which increases the overall monetary cost.

In the implementation of the *Find* and *Verify* stages in our approach, we explore two different crowdsourcing workflows combining different types of crowds. The first workflow combines LD experts and microtask workers: This workflow leverages the expertise of Linked Data experts in a contest to find and classify erroneous triples according to a pre-defined quality taxonomy, while the workers verify the outcome of the contest. The second workflow entirely relies on microtask crowdsourcing to perform the *Find* and *Verify* stages. As discussed in Section 3, these crowdsourcing approaches exhibit different characteristics in terms of the types of tasks they can be applied to, the way the results are consolidated and exploited, and the audiences they target. Therefore, in this work we study the impact on involving different types of crowd to detect quality issues in RDF triples: LD experts in the contest and workers in the microtasks. Table 1 presents a summary of the two approaches as they have been used in this work for LD quality assessment purposes.

Figure 1 depicts the steps carried out in each of the stages of the two crowdsourcing workflows studied in this work. In the following sections, we provide more details about the implementation of the variants of the *Find* and *Verify* stages.

4.1. Find Stage: Contest-based Crowdsourcing

In this implementation of the *Find* stage, we reached out to an expert crowd of researchers and Linked Data enthusiasts via a contest. The tasks in the contest consist on identifying and classifying specific types of Linked Data quality problems in DBpedia triples. To collect the contributions from this crowd, in previous work [43], we developed a web-based tool called *TripleCheckMate*[6] (cf. Figure 2). TripleCheckMate [22] allows users to select RDF resources, identify issues related to triples of the resource and classify these issues according to a pre-defined taxonomy of data quality problems [44]. A prize was announced for the

[6]`http://github.com/AKSW/TripleCheckMate`

Table 1

Comparison between the proposed approaches to crowdsource LD quality assessment.

Characteristic	**Contest-based**	**Microtask Crowdsourcing**
Participants	Controlled group: LD experts	Anonymous large group
Time duration	Long (weeks)	Short (days)
Reward	A final prize	Micropayments
Reward mechanism	"One participant gets it all": The contest winner gets the final prize.	"Each participant receives a payment": Each participant received a micropayment per solved task.
Tool/platform	*TripleCheckMate*	Amazon Mechanical Turk (MTurk)

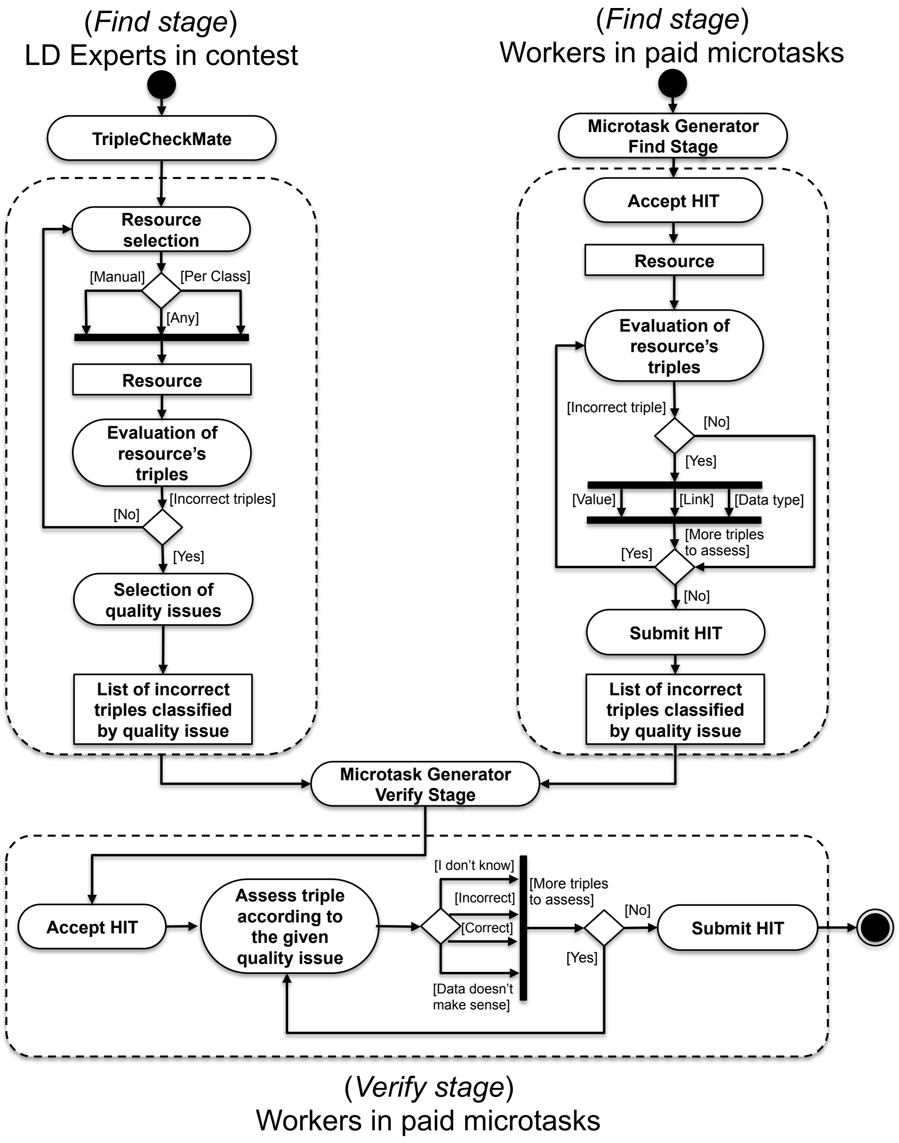
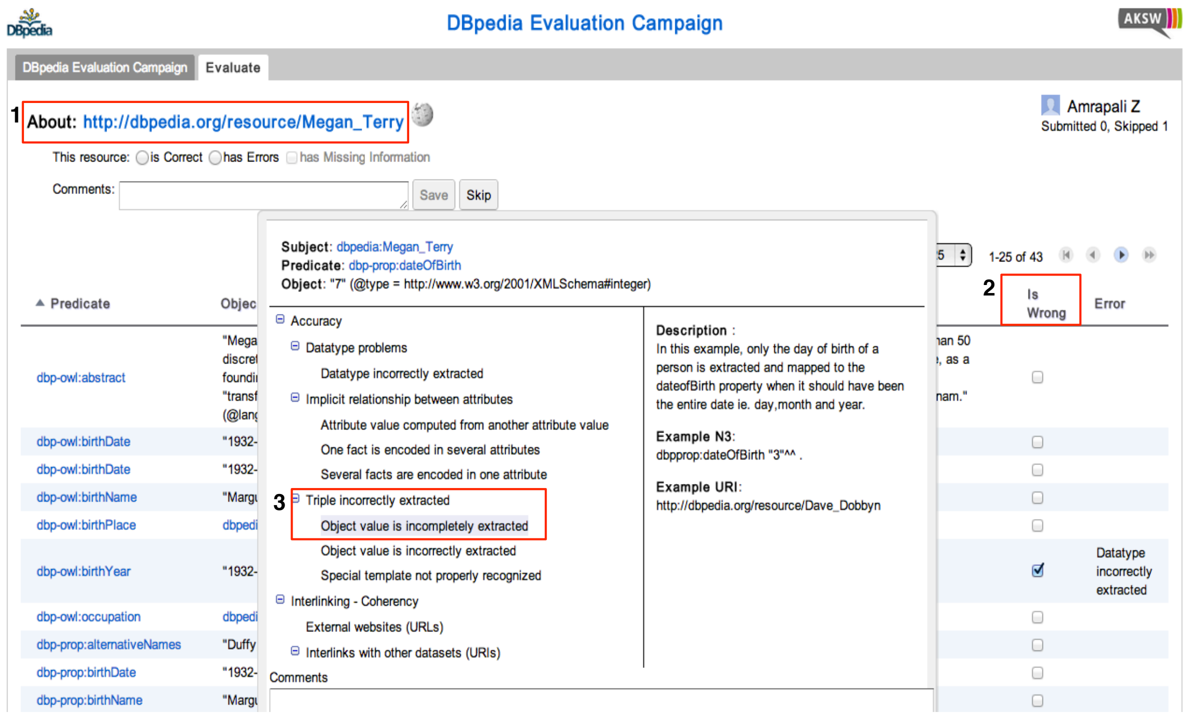

(Find stage)
LD Experts in contest
TripleCheckMate
Resource selection
[Manual]
[Per Class]
[Any]
Resource
Evaluation of resource's triples
[Incorrect triples]
[No]
[Yes]
Selection of quality issues
List of incorrect triples classified by quality issue
(Find stage)
Workers in paid microtasks
Microtask Generator Find Stage
Accept HIT
Resource
Evaluation of resource's triples
[Incorrect triple]
[No]
[Yes]
[Value]
[Link]
[Data type]
[More triples to assess]
[Yes]
[No]
Submit HIT
List of incorrect triples classified by quality issue
Microtask Generator Verify Stage
Accept HIT
Assess triple according to the given quality issue
[I don't know]
[Incorrect]
[Correct]
[Data doesn't make sense]
[More triples to assess]
[No]
[Yes]
Submit HIT
(Verify stage)
Workers in paid microtasks

Fig. 1. Workflow that combines LD experts with microtask crowdsourcing to perform LD quality assessment.


DBpedia Evaluation Campaign

1 About: http://dbpedia.org/resource/Megan_Terry

This resource: ◯ is Correct ◯ has Errors ☐ has Missing Information

Comments:

Subject: dbpedia:Megan_Terry
Predicate: dbp-prop:dateOfBirth
Object: "7" (@type = http://www.w3.org/2001/XMLSchema#integer)

- Accuracy
  - Datatype problems
    - Datatype incorrectly extracted
  - Implicit relationship between attributes
    - Attribute value computed from another attribute value
    - One fact is encoded in several attributes
    - Several facts are encoded in one attribute
  - 3 Triple incorrectly extracted
    - Object value is incompletely extracted
    - Object value is incorrectly extracted
    - Special template not properly recognized
- Interlinking - Coherency
  - External websites (URLs)
  - Interlinks with other datasets (URIs)

Comments

Description :
In this example, only the day of birth of a person is extracted and mapped to the dateofBirth property when it should have been the entire date ie. day,month and year.

Example N3:
dbpprop:dateOfBirth "3"^^ .

Example URI:
http://dbpedia.org/resource/Dave_Dobbyn

1-25 of 43

▲ Predicate	Objec		2 Is Wrong	Error
dbp-owl:abstract	"Mega discre foundi "transf (@lang	an 50 , as a nam."	☐	
dbp-owl:birthDate	"1932-		☐	
dbp-owl:birthDate	"1932-		☐	
dbp-owl:birthName	"Margu		☐	
dbp-owl:birthPlace	dbpedi		☐	
dbp-owl:birthYear	"1932-		☑	Datatype incorrectly extracted
dbp-owl:occupation	dbpedi		☐	
dbp-prop:alternativeNames	"Duffy		☐	
dbp-prop:birthDate	"1932-		☐	
dbp-prop:birthName	"Margu		☐	

Fig. 2. Screenshot of the *TripleCheckMate* crowdsourcing data quality assessment tool. (1) Displays the DBpedia resource that is currently assessed; (2) Users can specify that a triple is erroneous by checking the box ‘Is Wrong’; (3) Users select the corresponding quality issues present in the triple from a catalog, which contains a hierarchy of quality issues including detailed descriptions and examples for each issue.

user submitting the highest number of (real) quality problems.

The workflow starts when a user signs into the *TripleCheckMate* tool to participate in the contest, as shown in Figure 1. As a basic means to avoid spam, each user first has to login with his Google account through OAuth2. Then she is presented with three options to choose a resource from DBpedia: (i) ‘Any’, for random selection; (ii) ‘Per Class’, where she may choose a resource belonging to a particular class of her interest; and (iii) ‘Manual’, where she may provide a URI of a resource herself. Once a resource is selected following one of these alternatives, the user is presented with a table in which each row corresponds to an RDF triple of that resource. The next step is the actual quality assessment at triple level. The user is provided with the link to the corresponding Wikipedia page of the given resource in order to offer more context for the evaluation. If she detects a triple containing a problem, she checks the box ‘Is Wrong’. Moreover, she assigns specific quality problems (according to the classification devised in [43]) to troublesome triples, as depicted in Figure 2. The user can assess as many triples from a resource as desired, or select another resource to evaluate.

The *TripleCheckMate* tool only records the triples that are identified as ‘incorrect’. This is consistent with the definition of *Find* stage from the original *Find-Fix-Verify* pattern, where the crowd exclusively detects the problematic elements; while the remaining data is not taken into consideration. In addition, this tool measures inter-rater agreements. This means that DBpedia resources are typically checked multiple times. This redundancy mechanism is extremely useful to analyze the performance of the users (as we compare their responses against each other), to identify quality problems which are likely to be real (as they are confirmed by more than one opinion) and to detect unwanted behavior (as users are not ‘rewarded’ unless their assessments are ‘consensual’).

The outcome of this contest corresponds to a set of triples $\mathcal{T}$ judged as ‘incorrect’ by LD experts and classified according to the detected quality issues in $\mathcal{Q}$.

4.2. *Find Stage: Paid Microtask Crowdsourcing*

This *Find* stage applies microtasks solved by lay users from a crowdsourcing platform. In order to perform a fair comparison between the performance of LD experts and crowd workers, in this variant of the *Find* stage we aimed at implementing a similar workflow (including a similar user interface) for the crowd workers as the one provided to the LD experts. Therefore, in this stage the crowd is enquired for identifying quality issues on a set of RDF triples associated with RDF resources from the DBpedia data set. However, given that crowd workers are not necessarily knowledgeable about RDF or Linked Data, each microtask was augmented with human-readable information associated with the RDF triples. Formally, in our approach, a microtask is defined as follows.

Definition 4. (Microtask). *A microtask m is a set of 3-tuples (t, h_t, Q), where t is an RDF triple, h_t corresponds to human-readable information that describes t, and Q is the set of quality issues to be assessed on triple t.*

Following the MTurk terminology (cf. Section 3), each 3-tuple (t, h_t, Q) corresponds to a *question* while m is a *HIT (Human Intelligence Task)* with granularity (number of questions) equals to $|m|$.

The execution of this stage, as depicted in Figure 1, starts by generating the microtasks from $\mathcal{F}_i$, i.e., the sets of RDF triples $\mathcal{T}$ and quality issues $\mathcal{Q}$ to crowdsource. In addition, a parameter α can be specified as a threshold on the number of questions to include in a single microtask. Algorithm 1 presents the procedure to create the microtasks. The algorithm firstly performs a pruning step (line 2) to remove triples that do not require human assessment. For instance, in our implementation, the function $prune$ discards RDF triples whose URIs could not be dereferenced. The algorithm then proceeds to build microtasks such that each microtask only contains triples associated with a specific resource, similar to the interfaces of the *TripleCheckMate* tool used in the contest. The set $\mathcal{S}$ contains all the resources that appear as subjects in the set of triples $\mathcal{T}$ (line 3). For each subject, the algorithm builds the set of triples $\mathcal{T}'$ associated with the subject (line 5), and the creation of microtasks begins (line 6). From the pool $\mathcal{T}'$, a triple t is selected (line 8) and the corresponding human-readable information is extracted (line 9). In this stage, similar to the *TripleCheckMate*, each microtask requires the workers to browse all the possible quality issues, therefore, the set of issues to assess on triple t is equal to $\mathcal{Q}$ in each microtask created (line 10). In case that the number of questions in the current microtask exceeds the threshold α, a new microtask is then created. The definition of the parameter α allows for avoiding the creation of very long tasks, i.e., when the number of triples with the same subject is large; appropriate values of α enables the creation of tasks than can still be solved in a reasonable time, consistent with the concept of microtask (a short task). Algorithm 1 continues creating microtasks for all the triples of a resource (lines 7-16), for all the resources (lines 4-18). The outcome of the algorithm is a set $\mathcal{M}$ of microtasks to assess the quality of the triples in $\mathcal{T}$ according to the issues in $\mathcal{Q}$.

The generated microtasks are then submitted to the crowdsourcing platform. When a worker accepts a microtask or HIT, she is presented with a table that contains triples associated to an RDF resource, as shown in Figure 1. For each triple, the worker determines whether the triple is 'incorrect' with respect to a fixed set of quality issues $\mathcal{Q}$ (cf. Section 2): object incorrectly/incompletely extracted, datatype incorrectly extracted or incorrect link, abbreviated as 'Value', 'datatype', and 'Link', respectively. Once the worker has assessed all the triples within a microtask, she proceeds to submit the HIT. Consistently with the *Find* stage implemented with a contest, the outcome of the microtasks corresponds to a set of triples $\mathcal{T}$ judged as 'incorrect' by workers and classified according to the detected quality issues in $\mathcal{Q}$.

An important aspect when generating microtasks from RDF data (or machine-readable data in general) is developing useful human-understandable interfaces (Algorithm 1, line 9) for the target non-expert crowds. In microtasks, optimal user interfaces reduce ambiguity as well as the probability to retrieve erroneous answers from the crowd due to a misinterpretation of the task. Therefore, before start to resolve one of our tasks, the crowd workers were instructed with details and examples about each quality issue. After reading the instructions, workers proceed to resolve the given task. Figure 3 depicts the interface of a microtask generated for the *Find* stage in our approach. To display each triple, we retrieved the values of the foaf:name or rdfs:label properties for subjects, predicates, and datatypes. The name of languages in language-tagged strings were parsed using a conversion table from the best current practices BCP 47 [7], as suggested by the RDF specification[7]. Language tags and datatypes of objects were highlighted, such that workers can easily iden-

[7]http://www.w3.org/TR/rdf11-concepts/

Algorithm 1 Microtask Generator for Find Stage

Require: $\mathcal{F}_i = (\mathcal{T}, \mathcal{Q})$ and α, where $\mathcal{T}$ is a set of RDF triples, $\mathcal{Q}$ is the set of quality issues, α is the maximum number of triples grouped in a single microtask.
Ensure: A set of microtasks $\mathcal{M}$ to assess triples from $\mathcal{T}$ according to $\mathcal{Q}$.

```
1: $\mathcal{M} \leftarrow \emptyset$
2: $\mathcal{T}' \leftarrow prune(\mathcal{T})$
3: $\mathcal{S} \leftarrow \{s|(s,p,o) \in \mathcal{T}'\}$
4: for all $s \in \mathcal{S}$ do
5:     Build $T' \subseteq \mathcal{T}$ such that $T'' = \{t|t = (s,p,o) \wedge t \in T'\}$
6:     $m \leftarrow \emptyset$
7:     while $\mathcal{T}'' \neq \emptyset$ do
8:         Select a triple $t$ from $\mathcal{T}''$
9:         Extract human-readable information $h_t$ from RDF triple $t$
10:        $m \leftarrow m \cup \{(t, h_t, \mathcal{Q})\}$
11:        if $|m| \geq \alpha$ then
12:            $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
13:            $m \leftarrow \emptyset$
14:        end if
15:        $\mathcal{T}'' \leftarrow \mathcal{T}'' - \{t\}$
16:    end while
17:    $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
18: end for
19: return $\mathcal{M}$
```

tify them[8]. Additionally, in order to provide contextual information, we implemented a simple wrapper which extracts the corresponding data encoded in the infobox of the Wikipedia article associated with the resource – specified via foaf:isPrimaryTopicOf. The crowd has the possibility to select one or several quality issues per triple.

Further microtask design criteria related to spam detection and quality control; we used different mechanisms to discourage low-effort behavior which leads to random answers and to identify accurate answers (see Section 5.2.2).

4.3. Verify Stage: Paid Microtask Crowdsourcing

In this stage, we applied microtask crowdsourcing in order to verify quality issues in RDF triples identified

Algorithm 2 Microtask Generator for Verify Stage

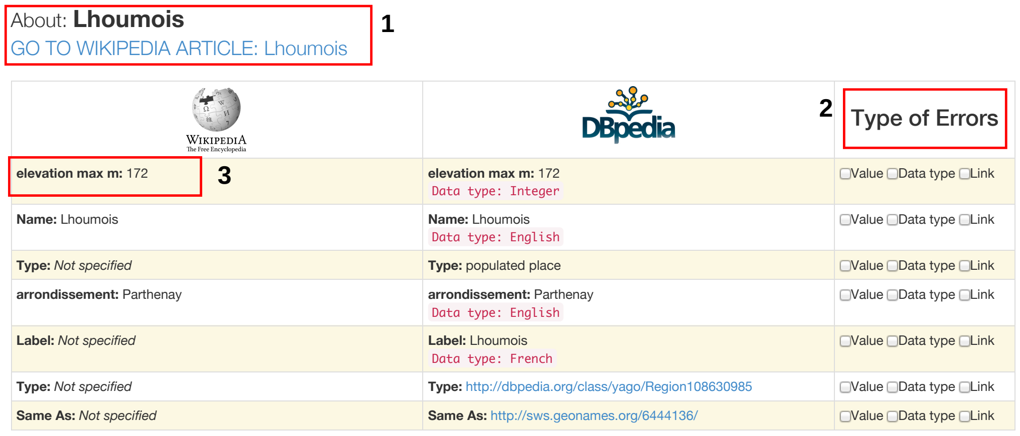
Require: $\mathcal{F}_o = (\mathcal{T}, \dot{\phi}(.))$ and β, where $\mathcal{T}$ is a set of RDF triples, $\dot{\phi}(.)$ is a mapping of triples in $\mathcal{T}$ to quality issues, and β is the maximum number of triples grouped in a single microtask.
Ensure: A set of microtasks $\mathcal{M}$ to assess triples from $\mathcal{T}$ annotated with quality issues $\ddot{\phi}(.)$.

```
1: $\mathcal{M}, \mathcal{F} \leftarrow \emptyset$
2: $\mathcal{T}' \leftarrow prune(\mathcal{T})$
3: $\mathcal{F}'_o \leftarrow (\mathcal{T}', \dot{\phi}(.))$ //$\mathcal{F}'_o$ contains non-pruned triples
4: for all $(t, \phi(t)) \in \mathcal{F}'_o$ do
5:     for all $q \in \phi(t)$ do
6:         $\mathcal{F} \leftarrow \mathcal{F} \cup \{(t,q)\}$
7:     end for
8: end for
9: $Q' \leftarrow \{q|(t,q) \in \mathcal{F}\}$
10: for all $q \in Q'$ do
11:    $m \leftarrow \emptyset$
12:    for all $(t,q) \in \mathcal{F}$ do
13:        Extract human-readable information $h_t$ from RDF triple $t$
14:        $m \leftarrow m \cup \{(t, h_t, q)\}$
15:        if $|m| \geq \beta$ then
16:            $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
17:            $m \leftarrow \emptyset$
18:        end if
19:    end for
20:    $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
21: end for
22: return $\mathcal{M}$
```

as problematic during the *Find Stage* (see Figure 1). To ensure that in this stage a proper validation is executed on each triple, the microtasks are simplified with respect to the ones from the *Find* stage: (i) each microtask focuses on a specific quality issue, (ii) the number of triples per microtask is reduced.

The generation of microtasks in this stage is presented in Algorithm 2. This algorithm groups the triples in $\mathcal{T}$ obtained from the previous stage by quality issue, which enables the workers to focus on one quality issue at the time. The input of this stage is the set of triples to assess $\mathcal{T}$ and their mappings to quality issues $\dot{\phi}(.)$. The parameter β specifies the number of questions to include in a single microtask. The algorithm firstly performs a pruning step (line 2) to remove certain triples. For instance, a triple t that was considered 'correct' in the *Find* stage ($\dot{\phi}(t) = \emptyset$) is discarded, consistently with the definition of the *Find-Fix-Verify*

[8]For the sake of simplicity, datatypes and language tags are introduced as a single issue to workers, therefore our interfaces display "datatype" even for language tags.



About: **Lhoumois**
GO TO WIKIPEDIA ARTICLE: Lhoumois

Wikipedia	DBpedia	Type of Errors
elevation max m: 172	**elevation max m:** 172 `Data type: Integer`	☐Value ☐Data type ☐Link
Name: Lhoumois	**Name:** Lhoumois `Data type: English`	☐Value ☐Data type ☐Link
Type: *Not specified*	**Type:** populated place	☐Value ☐Data type ☐Link
arrondissement: Parthenay	**arrondissement:** Parthenay `Data type: English`	☐Value ☐Data type ☐Link
Label: *Not specified*	**Label:** Lhoumois `Data type: French`	☐Value ☐Data type ☐Link
Type: *Not specified*	**Type:** http://dbpedia.org/class/yago/Region108630985	☐Value ☐Data type ☐Link
Same As: *Not specified*	**Same As:** http://sws.geonames.org/6444136/	☐Value ☐Data type ☐Link

Fig. 3. Screenshot of a microtask generated in the *Find* stage. (1) Displays the DBpedia resource that is currently assessed; (2) Users select the corresponding quality issues present in the triple; (3) Values extracted from the infobox of the Wikipedia article associated with the resource.

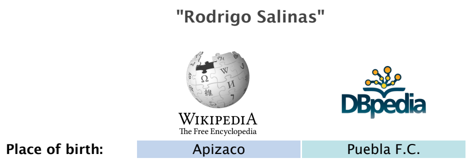
pattern [3]. Also, in our implementation, the function $prune$ discards answers whose inter-rater agreement were not higher than a certian value. The algorithm then proceeds to build microtasks such that each microtask only contains triples associated with a specific quality issue. For each answer from the previous stage, the algorithm decomposes the set of quality issues $\dot{\phi}(t)$ of a triple t into singletons (lines 3-7). The set $\mathcal{Q}$ contains all the quality issues present in the set of triples $\mathcal{T}$ (line 8). For each quality issue q (line 9), the algorithm processes all triples associated with that quality issue (lines 11-18). The algorithm extracts human-readable information about the triples (line 12) and appends it to the microtask (line 13). In case that the number of questions in the current microtask exceeds the threshold β, a new microtask is then created. The outcome of the algorithm is a set $\mathcal{M}$ of microtasks to assess the quality of the triples in $\mathcal{T}$ according to the issues identified in the *Find* stage ($\dot{\phi}(.)$).

Based on the classification of LD quality issues explained in Section 2, we created three different interfaces for the microtasks. Each microtask contains the description of the procedure to be carried out to complete the task successfully. We provided the worker examples of incorrect and correct examples along with four options (as shown in Figure 1): (i) ‘Correct’; (ii) ‘Incorrect’; (iii) ‘I cannot tell/I don’t know’; (iv) ‘Data doesn’t make sense’. The third option was meant to allow the user to specify when the question or values were unclear. The fourth option referred to those cases in which the presented data was truly unintelligible. Furthermore, the workers were not aware that the presented triples were previously identified as ‘incorrect’ in the *Find* stage and the questions were designed such that workers could not foresee the right answer. We describe the particularities of the interfaces of the microtask generated for the *Verify* stage in the following.

Incorrect/incomplete object value. In this type of microtask, we asked the workers to evaluate whether the value of a given RDF triple from DBpedia is correct or not. We displayed human-readable information retrieved by dereferencing the URIs of the subject and predicate of the triple. In particular, we selected the values of the foaf:name or rdfs:label properties for each subject and predicate. Additionally, we extracted the values from the infobox of the Wikipedia article associated with the subject of the triple using the wrapper implemented in the *Find* stage (cf. Section 4.2). Figure 4 depicts the interface of the resulting tasks.

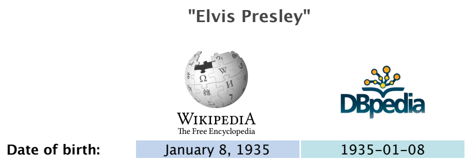
In the task presented in Figure 4a, the worker must decide whether the place of birth of “Rodrigo Salinas” is correct. According to the DBpedia triple, the value of this property is Puebla F.C, while the information extracted from Wikipedia, suggests that the right value is Apizaco. Therefore, the right answer to this tasks is: the DBpedia data is incorrect.

An example of a DBpedia triple whose value is correct is depicted in Figure 4b. In this case, the worker must analyze the date of birth of “Elvis Presley”. According to the information extracted from Wikipedia, the date of birth of Elvis Presley is January 8, 1935, while the DBpedia value is 1935-01-08. Despite the dates are represented in different formats, semantically the dates are indeed the same, thus the DBpedia value is correct.


"Rodrigo Salinas"

	Wikipedia	DBpedia
Place of birth:	Apizaco	Puebla F.C.

(a) **Incorrect** object value in DBpedia


"Elvis Presley"

	Wikipedia	DBpedia
Date of birth:	January 8, 1935	1935-01-08

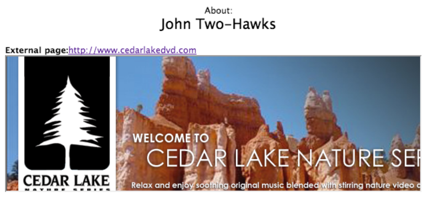
(b) **Correct** object value in DBpedia

Fig. 4. Incorrect/incomplete object value: The crowd must compare the DBpedia and Wikipedia values and decide whether the DBpedia entry is correct or not for a given subject and predicate.

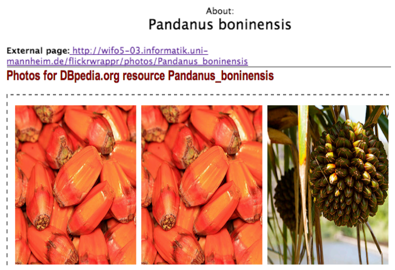
Incorrect datatypes & language tags. This type of microtask consists of detecting those DBpedia triples whose object datatype or language tags were not correctly assigned. The generation of the interfaces for these tasks was very straightforward, by dereferencing the URIs of the subject and predicate of each triple and displaying the values for the foaf:name or rdfs:label.

In the description of the task, we introduced the crowd the concept of data type of a value and provided two simple examples. The first example illustrates when the language tag (rdf:langString) is incorrect while analyzing the entity "Torishima Izu Islands": *Given the property "name", is the value "鳥島" of type "English"?* A worker does not need to understand that the name of this island is written in Japanese, since it is evident that the language type "English" in this example is incorrect. In a similar fashion, we provided an example where the language tag is assigned correctly by looking at the entity "Elvis Presley": *Given the property "name", is the value "Elvis Presley" of type "English"?* According to the information from DBpedia, the value of the name is written in English and the type is correctly identified as English.

Incorrect links. In this type of microtask, we asked the workers to verify whether the content of the external page referenced from the Wikipedia article corresponds to the subject of the RDF triple. For the interface of the HITs, we provided the worker a preview of the Wikipedia article and the external page by implementing HTML iframe tags. In addition, we retrieved the foaf:name of the given subject and the link to the corresponding Wikipedia article using the predicate foaf:isPrimaryTopicOf.


About:
John Two-Hawks
External page: http://www.cedarlakedvd.com

(a) External link displaying **unrelated** content to the subject


About:
Pandanus boninensis
External page: http://wifo5-03.informatik.uni-mannheim.de/flickrwrappr/photos/Pandanus_boninensis
Photos for DBpedia.org resource Pandanus_boninensis

(b) Web page displaying **related** images to the subject

Fig. 5. Incorrect link: The crowd must decide whether the content from an external web page is related to the subject.

Examples of this type of task are depicted in Figure 5. In the first example (see Figure 5a), the workers must decide whether the content in the given external web page is related to "John Two-Hawks". It is easy to observe that in this case the content is not directly associated to the person "John Two-Hawks". Therefore, the right answer is that the link is incorrect. On the other hand, we also exemplified the case when an interlink presents relevant content to the given subject. Consider the example in Figure 5b, where the subject is the plant "Pandanus boninensis" and the external link is a web page generated by the DBpedia Flickr wrapper. The web page indeed shows pictures of the subject plant. Therefore, the correct answer is that the link is correct.

4.4. *Properties of Our Approach*

Given that the contest settings are handled through the *TripleCheckMate* tool, in this section we expose the properties of the proposed microtask crowdsourcing approaches. First, we demonstrate that the algorithms for microtask generation in the *Find* and *Verify* stages are efficient in terms of time.

Proposition 1. *The time complexity of the microtask generators is $\mathcal{O}(|\mathcal{T}|)$ for the Find stage and $\mathcal{O}(|\mathcal{T}||\mathcal{Q}|)$ for the Verify stage.*

Proof. *The algorithm of the Find stage iterates over all the triples associated with each distinct triple subject in $\mathcal{T}$, therefore the complexity of this stage is $\mathcal{O}(|\mathcal{T}|)$. In the Verify stage, the algorithm firstly iterates over the answers obtained from the previous stage, which corresponds to $\mathcal{T}$. Next, the algorithm iterates over the quality issues detected in the Find stage; in the worst case, each quality issue is found in at least one triple, then, the set Q' is equal to $\mathcal{Q}$. For each quality issue, the algorithm processes the triples annotated with that quality issue, which again in the worst case is $\mathcal{T}$ (all the triples present all the quality issues). Therefore, the complexity of the Find stage is calculated as $\mathcal{O}(|\mathcal{T}| + |\mathcal{T}||\mathcal{Q}|)$, then $\mathcal{O}(|\mathcal{T}||\mathcal{Q}|)$.* ■

One important aspect when applying paid microtask crowdsourcing is the number of generated tasks, since this determines the overall monetary cost. The following theorem states the efficiency of Algorithms 1 and 2 in terms of the number of crowdsourced microtasks.

Proposition 2. *The number of microtasks generated in each stage is linear with respect to the number of triples assessed.*

Proof. *In the Find stage, a microtask is generated when the number of triples within task exceeds the threshold α. Since in this stage each microtask groups triples by subjects, then the number of microtaks per subject is given by $\left\lceil \frac{|\{(p,o)|(s_i,p,o)\in\mathcal{T}\}|}{\alpha} \right\rceil$, where $\{(p,o)|(s_i,p,o) \in \mathcal{T}\}$ corresponds to triples with subject s_i. In total, in the Find stage, the exact number of microtasks generated is $\sum_{s_i\in\mathcal{S}} \left\lceil \frac{|\{(p,o)|(s_i,p,o)\in\mathcal{T}\}|}{\alpha} \right\rceil$, which is less than $|\mathcal{T}|$ (for α >1). In the Verify stage, each microtask groups RDF triples with the same quality issue. When considering β as the maximum number of triples contained within a microtask, then the number of microtasks created per quality issue $q_i \in \mathcal{Q}$ is $\left\lceil \frac{|\{t|t\in\mathcal{T} \,\wedge\, q_i\in\dot{\phi}(t)\}|}{\beta} \right\rceil$. Therefore, the exact number of microtasks generated in the Verify stage is $\sum_{q_i\in\mathcal{Q}} \left\lceil \frac{|\{t|t\in\mathcal{T} \,\wedge\, q_i\in\dot{\phi}(t)\}|}{\beta} \right\rceil$, which is $\leq |\mathcal{T}||\mathcal{Q}|$. Considering that the set $\mathcal{Q}$ is considerably smaller than $\mathcal{T}$, we can affirm that the number of microtasks generated in the Verify stage is linear with respect to $\mathcal{T}$.* ■

When analyzing the number of microtasks generated in each stage, the *Verify* stage seems to produce more tasks than the *Find* stage. This is a consequence of simplifying the difficulty of the microtasks in the *Verify* stage, where workers have to assess only one type of quality issue at the time. However, in practice, the number of microtasks generated in the *Verify* stage is not necessarily larger. For instance, in our experiments with LD experts and crowd workers, we observed that large portions of the triples are not annotated with quality issues in the *Find* stage. Since Algorithm 2 prunes triples with no quality issues (consistently with the definition of the *Find-Fix-Verify* pattern), the subset of triples crowdsourced in the *Verify* stage is considerably smaller than the original set, hence the number of microtasks to verify is reduced.

A summary of our microtask crowdsourcing approach implemented for the *Find* and *Verify* stages is presented in Table 2.

5. Evaluation

We empirically analyzed the performance of the two crowdsourcing workflows described in Section 4: the first workflow combines LD experts in the *Find* stage with microtask (lay) workers from MTurk in the *Verify* stage; the second workflow consists of executing both *Find* and *Verify* stages with microtask workers. It is important to highlight that, in the experiments of the *Verify* stage, workers did not know that the data provided to them was previously classified as problematic.

In addition, we executed baseline approaches to detect quality issues which allow us to understand the strengths and limitations of applying crowdsourcing in this scenario. We used RDFUnit [21] for 'object value' and 'datatype' issues, and implemented a simple baseline for detecting incorrect 'interlinks'.

5.1. Experimental Settings

5.1.1. Dataset and Implementation

In our experiments, the assessed triples were extracted from the DBpedia dataset (version 3.9).[9] As described in Section 4.1, the *TripleCheckMate* tool was used in the contest. For the microtask crowdsourcing approaches, Algorithms 1 and 2 were implemented to generate the corresponding microtasks for the *Find* and *Verify* stages, respectively, in Python 2.7.2. Resulting microtasks were submitted as HITs to Amazon Mechanical Turk using the MTurk SDK for Java.[10]

[9]`http://wiki.dbpedia.org/Downloads39`
[10]`http://aws.amazon.com/code/695`

Table 2

Comparison between the Find and Verify stages in our approach. $\mathcal{T}$ is the set of RDF triples subject to crowdsourcing in each stage; $\mathcal{Q}$ corresponds to the set of quality issues; $\mathcal{S}$ is the set of distinct subjects of the triples in $\mathcal{T}$; α, β are the parameters that define the number of questions per microtask in the *Find* and *Verify* stage, respectively.

Characteristic	**Find Stage**	**Verify Stage**
Goal per task	Detecting and classifying LD quality issues in RDF triples.	Confirming LD quality issues in RDF triples.
Task generation complexity	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert\lvert\mathcal{Q}\rvert)$
Total tasks generated (only for microtask crowdsourcing)	$\sum_{s_i \in \mathcal{S}} \left\lceil \frac{\lvert\{(p,o) \mid (s_i,p,o) \in \mathcal{T}\}\rvert}{\alpha} \right\rceil$	$\sum_{q_i \in \mathcal{Q}} \left\lceil \frac{\lvert\{t \mid t \in \mathcal{T} \wedge q_i \in \dot{\phi}(t)\}\rvert}{\beta} \right\rceil$
Task difficulty	*High:* Each task requires knowledge on data quality issues; participants have to browse large number of triples.	*Medium-low:* Each task consists of validating pre-processed and classified triples; each task focuses in one

5.1.2. Metrics

The goal of our experiments is to detect whether RDF triples are incorrect. Based on this, we define:

– True Positive (TP): Incorrect triple classified as *incorrect*.
– False Positive (FP): Correct triple classified as *incorrect*.
– True Negative (TN): Correct triple classified as *correct*.
– False Negative (FN): Incorrect triple classified as *correct*.

To measure the performance of the studied crowdsourcing approaches (contest and microtasks), we report on: i) *inter-rater agreement* computed with the Fleiss' kappa metric in order to measure the consensus degree among raters (experts or MTurk workers); ii) *precision* to measure the quality of the outcome of each crowd, computed as $\frac{TP}{TP+FP}$.

5.1.3. Gold Standard

Two of the authors of this paper (MA, AZ) generated a gold standard for two samples of the crowdsourced triples. To generate the gold standard, each author independently evaluated the triples. After an individual assessment, they compared their results and resolved the conflicts via mutual agreement. The first sample evaluated corresponds to the set of triples obtained from the contest and submitted to MTurk. The inter-rater agreement between the authors for this first sample and was 0.4523 for object values, 0.5554 for datatypes, and 0.5666 for interlinks. For the second sample, we analyzed a subset from the triples identified in the *Find* stage by the crowd as 'incorrect'. The subset has the same distribution of quality issues and triples as the one assessed in the first sample: 509 triples for object values, 341 for datatypes/language tags, and 223 for interlinks. We measured the inter-rater agreement for this second sample and was 0.6363 for object values, 0.8285 for datatypes, and 0.7074 for interlinks. The inter-rater agreement values were calculated using the Cohen's kappa measure [6], designed for measuring agreement among two annotators. Disagreement arose in the object value triples when one of the reviewers marked number values which are rounded up to the next round number as correct. For example, the length of the course of the "1949 Ulster Grand Prix" was 26.5Km in Wikipedia but rounded up to 27Km in DBpedia. In case of datatypes, most disagreements were considering the datatype "number" of the value for the property "year" as correct. For the links, those containing unrelated content, were marked as correct by one of the reviewers since the link existed in the Wikipedia page.

The tools used in our experiments and the results are available online, including the outcome of the contest,[11] the gold standard and microtask data (HITs and results).[12]

5.2. Evaluation of Combining LD Experts (Find Stage) and Microtasks (Verify Stage)

5.2.1. Contest Settings: Find Stage

Participant expertise: We relied on the expertise of members of the Linked Data and the DBpedia communities who were willing to take part in the contest.

[11]`http://nl.dbpedia.org:8080/TripleCheckMate/`
[12]`http://people.aifb.kit.edu/mac/DBpediaQualityAssessment/`

Task complexity: In the contest, each participant was assigned the concise bound description of a DBpedia resource. All triples belonging to that resource were displayed and the participants had to validate each triple individually for quality problems. Moreover, when a problem was detected, the participant had to map it to one of the problem types from a quality problem taxonomy.
Monetary reward: We awarded the participant who evaluated the highest number of resources a Samsung Galaxy Tab 2 worth 300 EU.
Assignments: Each resource was evaluated by at most two different participants.

5.2.2. *Microtask Settings: Verify Stage*

Worker qualification: In MTurk, the requester can filter workers according to different qualification metrics. In this experiment, we recruited workers with "Approval Rate" greater than 50%.
HIT granularity: In each HIT, we asked the workers to solve five different questions ($\beta = 5$). Each question corresponds to an RDF triple and each HIT contains triples classified into one of the three quality issue categories discussed earlier.
Monetary reward: The micropayments were fixed to 4 US dollar cents. Considering the HIT granularity, we paid 0.04 US dollar per 5 triples.
Assignments: The number of assignments was set up to five and the answer was selected applying majority voting. We additionally compared the quality achieved by a group of workers vs. the resulting quality of the worker who submitted the first answer, in order to test whether paying for more answers (assignments) actually increases the quality of the results.

5.2.3. *Overall Results*

The contest was open for a predefined period of time of three weeks. During this time, 58 LD experts analyzed 521 distinct DBpedia resources and, considering an average of 47.19 triples per resource in this data set [43], the experts browsed around $24,560$ triples. They detected a total of $1,512$ triples as erroneous and classified them using the given taxonomy. After obtaining the results from the experts, we filtered out duplicates, triples whose objects were broken links and the external pages referring to the DBpedia Flickr Wrapper. In total, we submitted $1,073$ triples to the crowd. A total of 80 distinct workers assessed all the RDF triples in four days. A summary of these observations are shown in Table 3.

We compared the common $1,073$ triples assessed in each crowdsourcing approach against our gold standard and measured precision as well as inter-rater agreement values for each type of task (see Table 4). For the contest-based approach, the tool allowed two participants to evaluate a single resource. In total, there were 268 inter-evaluations for which we calculated the triple-based inter-agreement (adjusting the observed agreement with agreement by chance) to be 0.38. For the microtasks, we measured the inter-rater agreement values between a maximum of 5 workers for each type of task using Fleiss' kappa measure [10]. While the inter-rater agreement between workers for the interlinking was high (0.7396), the ones for object values and datatypes was moderate to low with 0.5348 and 0.4960, respectively. Table 4 reports on the precision achieved by the LF experts and crowd in each stage. In the following we present further details on the results for each type of task.

5.2.4. *Results: Incorrect/missing Values*

As reported in Table 4, our crowdsourcing experiments reached a precision of 0.90 for MTurk workers (majority voting) and 0.72 for LD experts. Most of the missing or incomplete values that are extracted from Wikipedia occur with the predicates related to dates, for example: `(2005 Six Nations Championship, Date, 12)`. In these cases, the experts and workers presented a similar behavior, classifying 110 and 107 triples correctly, respectively, out of the 117 assessed triples for this class. The difference in precision between the two approaches can be explained as follows. There were 52 DBpedia triples whose values might seem erroneous, although they were correctly extracted from Wikipedia. One example of these triples is: `(English (programming language), Influenced by, ?)`. We found out that the LD experts classified all these triples as incorrect. In contrast, the workers successfully answered that 50 out of this 52 were correct, since they could easily compare the DBpedia and Wikipedia values in the HITs.

5.2.5. *Results: Incorrect Datatypes or Language Tags*

Table 4 exhibits that the experts are reliable (with 0.83 of precision) on *finding* this type of quality issue, while the precision of the crowd (0.51) on *verifying* these triples is relatively low. In particular, the first answers submitted by the crowd were slightly better than the results obtained with majority voting. A detailed study of these cases showed that 28 triples that were initially classified correctly, later were misclassified, and most of these triples refer to a language datatype. The low performance of the MTurk workers compared to the experts is not surprising, since

Table 3

Overall results in each type of crowdsourcing approach.

	Contest-based	**Paid microtasks**
Number of distinct participants	Total: 58	Object values: 35 Datatypes/Language tags: 31 Interlinks: 31 Total: 80
Total no. of microtasks generated	–	216
Total time	3 weeks (predefined)	4 days
Total no. of triples evaluated	1,512	1,073
Object values	550	509
Datatype/Language tags	363	341
Interlinks	599	223

Table 4

Inter-rater agreement and precision values achieved with the implemented approaches.

Stage and Crowd	**Object values**	**Datatypes/Language Tags**	**Interlinks**
		Inter-rater agreement	
Find: LD experts		Calculated for all the triples: 0.38	
Verify: MTurk workers	0.5348	0.4960	0.7396
		(True positives, False positives)	
Find: LD experts	(364, 145)	(282, 59)	(34, 189)
Verify: MTurk workers (first answer)	(257, 108)	(144, 138)	(21, 13)
Verify: MTurk workers (majority voting)	(307, 35)	(134, 148)	(32, 2)
		Achieved precision	
Find: LD experts	0.7151	0.8270	0.1525
Verify: MTurk workers (first answer)	0.7041	0.5106	0.6176
Verify: MTurk workers (majority voting)	0.8977	0.4752	0.9412

this particular task requires certain technical knowledge about datatypes and, moreover, the specification of values and types in LD.

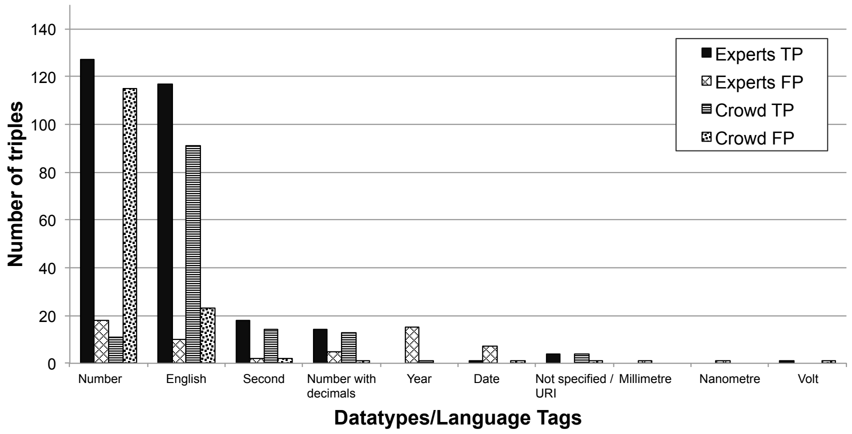
In order to understand the previous results, we analyzed the performance of experts and workers at a more fine-grained level. We calculated the frequency of occurrences of datatypes in the assessed triples (see Figure 6a) and reported the number of true positives (TP) and false positives (FP) achieved by both crowdsourcing methods for each type of task. Figure 6b depicts these results. The most notorious result in this task is the assessment performance for the datatype "number". The experts effectively identified triples where the datatype was incorrectly assigned as "number"[13] for instance, in the triple `(Walter Flores, date of birth, 1933)` the value `1933` was number instead of date. These are the cases where the crowd was confused and determined that datatype was correct, thus generating a large number of false positives. Nevertheless, it could be argued that the data type "number" in the previous example is not completely incorrect, when being unaware of the fact that there are more specific data types for representing time units. Under this assumption, the precision of the crowd would have been 0.8475 and 0.8211 for first answer and majority voting, respectively.

While looking at the language-tagged strings in "English" (in RDF `@en`), Figure 6b shows that the experts perform very well when discerning whether a given value is an English text or not. The crowd was less successful in the following two situations: (i) the value corresponded to a number and the remaining data was specified in English, e.g., `(St. Louis School Hong Kong, founded, 1864)`; and (ii) the value was a

[13]This error is very frequent when extracting dates from Wikipedia as some resources only contain partial data, e.g., only the year is available and not the whole date.

(a) Frequency of datatypes in the crowdsourced triples.

Datatype/Language Tag	Frequency	Datatype/Language Tag	Frequency
Number	145	Date	19
English	127	Not specified/URI	20
Second	20	Millimetre	1
Number with decimals	19	Nanometre	1
Year	15	Volt	1



(b) True positives (TP) and false positives (FP) per datatype in each crowdourcing method.

Fig. 6. Analysis of true and false positives in "Incorrect datatype" task for the first crowdsourcing workflow.

text without special characters, but in a different language than English, for example German (`Woellersdorf -Stein abrueckl, Art, Marktgemeinde`). The performance of both crowdsourcing approaches for the remaining datatypes were similar or not relevant due the low number of triples processed.

5.2.6. Results: Incorrect Links

Table 4 displays the precision for each studied quality assessment mechanism. The extremely low precision of 0.15 of the contest's participants was unexpected. We analyzed in detail the 189 misclassifications of the experts:

– The 95 Freebase links[14] connected via `owl:sameAs` were marked as incorrect, although both the subject and the object were referring to the same real-world entity.
– There were 77 triples whose objects were Wikipedia-upload entries; 74 of these triples were also classified incorrectly.
– 20 links (blogs, web pages, etc.) referenced from the Wikipedia article of the subject were also misclassified, regardless of the language of the content in the web page.

[14]`http://www.freebase.com`

The two settings of the MTurk workers outperformed the baseline approach. The 'first answer' setting reports a precision of 0.62, while the 'majority voting' achieved a precision of 0.94. The 6% of the links that were not properly classified by the crowd corresponds to those web pages whose content is in a different language than English or, despite they are referenced from the Wikipedia article of the subject, their association to the subject is not straightforward. Examples of these cases are the following subjects and links: 'Frank Stanford' and `http://nw-ar.com/drakefield/`, 'Forever Green' and `http://www.stirrupcup.co.uk`. We hypothesize that the design of the user interface of the HITs – displaying a preview of the web pages to analyze – helped the workers to easily identify those links containing related content to the triple subject.

5.3. *Evaluation of Using Microtrask Crowdsourcing in Find and Verify Stages*

5.3.1. Microtask Settings: Find and Verify Stages

The microtasks crowdsourced in the Find stage were configured as follows:

Worker qualification: We recruited workers whose "Approval Rate" is greater than 50%.

HIT granularity: In each HIT, we asked the workers to assess a maximum of 30 different triples with the same subject ($\alpha = 30$).
Monetary reward: The micropayments were fixed to 6 US dollar cents.
Assignments: The assignments were set up to 3 and we applied majority voting to aggregate the answers.

All triples identified as erroneous by at least to workers in the *Find* stage were candidates for crowdsourcing in the *Verify* stage. The microtasks generated in the subsequent stage were crowdsourced with the exact same configurations used in the *Verify* stage from the first workflow (cf. Section 5.2.2).

5.3.2. *Overall Results*

In order to replicate the approach followed in the contest, in the *Find* stage, we crowdsourced all the triples associated with resources that were explored by the LD experts. In total, we submitted to the crowd $30,658$ RDF triples. The microtasks were resolved by 187 distinct workers who identified $26,835$ triples as erroneous in 14 days, and classified them into the three quality issues studied in this work. Then, we selected samples from triples identified as erroneous in the *Find* stage by at least two workers from the crowd. This allowed us to fairly compare the outcome of the *Verify* stage from both workflows. Each sample contains the exact same number of triples that were crowdsourced in the Verify Stage in the first workflow, i.e., 509 triples with object value issues, 341 with data type or language tag issues, and 223 with interlinks issues. All triples crowdsourced in the *Verify Stage* were assessed by 141 workers in seven days. A summary of these results and further details are presented in Table 5.

Similar to the previous experiment, we measured the inter-rater agreement achieved by the crowd in both stages using the Fleiss' kappa metric. In the *Find* stage the inter-rater agreement of workers was 0.2695, while in the *Verify* stage, the the crowd achieved substantial agreement for all the types of tasks: 0.6300 for object values, 0.7957 for data types or language tags, and 0.7156 for interlinks. In comparison to the first workflow, the crowd in the *Verify* stage achieved higher agreement. This suggests that the triples identified as erroneous in the *Find* stage were easier to interpret or process by the crowd. Table 6 reports on the precision achieved by the crowd in each stage. It is important to notice that in this workflow we crowdsourced all the triples that could have been explored by the LD experts in the contest. In this way, we evaluate the performance of lay user and experts under similar conditions. During the *Find* stage, the crowd achieved low values of precision for the three types of tasks, which suggests that this stage is still very challenging for lay users. In the following we present further details on the results for each type of task.

5.3.3. *Results: Incorrect/missing Values*

In the *Find* stage, the crowd achieved a precision of 0.3713 for identifying 'incorrect/missing values', as reported in Table 6. In the following we present relevant observations derived form this evaluation:

- 46 false positives were generated for triples with predicates corresponding to dbpedia-prop:placeOfBirth, and dbpedia-prop:dateOfBirth, although for some of them the value extracted from Wikipedia coincided with the DBpedia value.
- 22 triples identified as 'incorrect' by the crowd encode metadata about the DBpedia extraction framework via predicates like dbpedia-owl:wikiPageID and dbpedia-owl:wikiPageRevisionID. This is a clear example in which a certain level of expertise in Linked Data (especially DBpedia) plays an important role in this task, since it is not straightforward to understand the meaning of these type of predicates. Furthermore, given the fact that triples with reserved predicates do not require further validation,[15] these triples could be entirely precluded from any crowd-based assessment.
- In 24 false positives, the human-readable information (label) extracted for triple predicates were not entirely comprehensible, e.g., "longd", "longs", "longm", "refnum", "sat chan", among others. This could negatively impact the crowd performance, since workers rely on RDF resource descriptions to discern whether triples values are correct or not.
- 14 triples encoding geographical coordinates via the predicates geo:lat, geo:long, and grs:point[16] were misinterpreted by the crowd as the values of these predicates were incorrect. This is because in DBpedia coordinates are represented as decimals, e.g., (dbpedia:Salasco, geo:lat, 45.3333), while in Wikipedia coordinates are represented using a Geodetic system, e.g., "Salasco latitude 45°20'N".

[15]DBpedia triples whose predicates are defined as "Reserved for DBpedia" should not be modified, since they encode special metadata generated during the extraction process.

[16]Prefixes geo and grs correspond to `http://www.w3.org/2003/01/geo/wgs84_pos#lat` and `http://www.georss.org/georss/point`, respectively.

Table 5

Overall results in each type of crowdsourcing approach in first crowdsourcing workflow: combing LD experts and microtask workers.

	Paid microtasks: Find stage	**Paid microtasks: Verify stage**
Number of distinct participants	Total: 187	Object values: 77 Datatypes/Language tags: 29 Interlinks: 46 Total: 141
Total no. of triples crowdsourced	68,976	1,073
Total no. of microtasks generated	2,339	216
Total time	14 days	7 days
Total no. of triples evaluated	26,835	1,073
Object values	8,691	509
Datatypes/Language tags	13,194	341
Interlinks	13,732	223

Table 6

Inter-rater agreement and precision values achieved in the first crowdsourcing workflow: combining LD experts and microtask workers.

Stage and Crowd	**Object values**	**Datatypes/Language Tags**	**Interlinks**
		Inter-rater agreement	
Find: MTurk workers		Calculated for all the triples: 0.2695	
Verify: MTurk workers	0.6300	0.7957	0.7156
		(True Positives, False Positives)	
Find: MTurk workers	(189, 320)	(50, 291)	(54, 169)
Verify: MTurk workers (first answer)	(126, 127)	(27, 22)	(39, 76)
Verify: MTurk workers (majority voting)	(175, 170)	(41, 6)	(53, 101)
		Achieved Precision	
Find: MTurk workers	0.3713	0.1466	0.2422
Verify: MTurk workers (first answer)	0.4980	0.5510	0.3391
Verify: MTurk workers (majority voting)	0.5072	0.8723	0.3442

The crowd in the *Verify* stage achieved similar precision for both settings ‘first answer’ and ‘majority voting’, with values of 0.4980 and 0.5072, respectively. Errors from the first iteration were reduced in the *Verify* stage, especially in triples with predicates dbpedia-prop:dateOfBirth and dbpedia-prop:placeOfBirth; 38 out of 46 of these triples were correctly classified in the *Verify* stage. Workers in this stage still made similar errors as the ones previously discussed – triples encoding DBpedia metadata and geo-coordinates, and incomprehensible predicates – although in a lower scale in comparison to the *Find* stage.

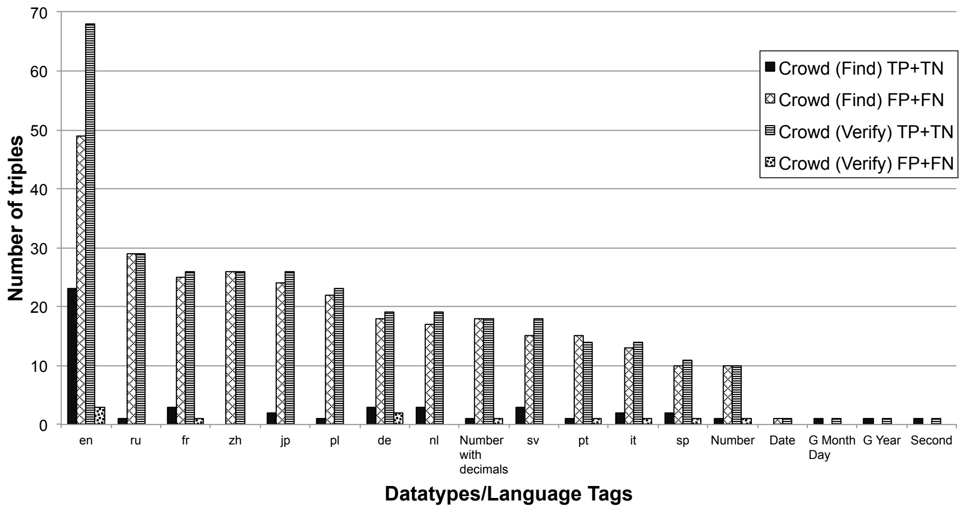
5.3.4. Results: Incorrect Datatypes or Language Tags

In this type of task, the crowd in the *Find* stage focused on assessing triples whose objects correspond to language-tagged literals. Figure 7a shows the distribution of the datatypes and language tags in the sampled triples processed by the crowd. Out of the 341 analyzed triples, 307 triples identified as ‘erroneous’ in this stage were annotated with language tags. As reported on Table 6, the crowd in the *Find* stage achieved a precision of 0.1466, being the lowest precision achieved in all the microtask settings. Most of the triples (72 out of 341) identified as ‘incorrect’ in this stage were annotated with the English language tag. We corroborated that false positives in other languages were not generated due to malfunctions of the interface of the HITs: microtasks were properly displaying non UTF-8 characters used in several languages in DBpedia, e.g., Russian, Japanese, Chinese, among others.

In the *Verify* stage of this type of task, the crowd outperformed the precision of the *Find stage*, achieving values of 0.5510 for the ‘first answer’ setting and 0.8723 with ‘majority voting’. This major improvement on the precision put in evidence the importance of having the a multi-validation pattern like *Find-Fix-*

(a) Frequency of datatypes in the crowdsourced triples.

Datatype/Language Tag	Frequency	Datatype/Language Tag	Frequency
English (en)	72	Swedish (sv)	18
Russian (ru)	30	Portuguese (pt)	16
French (fr)	20	Italian (it)	15
Chinese (zh)	26	Spanish; Castilian (es)	12
Japanese (jp)	26	Number	11
Polish (pl)	23	Date	1
German (de)	21	G Month	1
Dutch; Flemish (nl)	20	G Year	1
Number with decimals	19	Second	1



(b) True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) per datatype in each stage (*Find*, *Verify*).

Fig. 7. Analysis of results in the "Incorrect datatype" task for second crowdsourcing workflow based on microtasks.

Verify in which initial errors can be reduced in subsequent iterations. Congruently with the behavior observed in the first workflow, MTurk workers perform well when verifying language-tagged literals. Furthermore, the high values of inter-rater agreement confirms that the crowd is consistently good in this particular scenario. Figure 7b depicts the results of the 'majority voting' setting when classifying triples correctly, i.e., true positives (TP) and true negatives (TN), vs. misclassifying triples, i.e. false positives (FP) and false negatives (FN). We can observe that the crowd is exceptionally successful in identifying correct triples that were classified as erroneous in the previous stage (true negatives). This can be confirmed by the high value of accuracy[17] (0.9531) achieved by the crowd in this stage with 'majority voting'. A closer inspection to the six false positives revealed that in three cases the crowd misclassified triples whose object is a proper noun, for instance, (Tiszaszentimre, name, Tiszaszentimre@en) and (Ferrari Mythos, label, Ferrari Mythos@de); in the other three cases the object of the triple corresponds to a common noun or text in the following languages: Italian, Portuguese, and English, for example, (Book, label, Libro@it).

5.3.5. *Results: Incorrect Links*

From the studied sample, in the *Find* stage the crowd classified as 'incorrect interlink', those RDF triples whose objects correspond to RDF resources (and not Web pages); this is the case of the majority of the triples. We analyzed in detail the characteristics of the 169 misclassified triples by the crowd in this stage:

– Out of the 223 triples analyzed, the most popular predicate corresponds to rdf:type (found in 167 triples). For this predicate, the crowd misclassified 114 triples. The majority of the ob-

[17]Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$

jects of these triples correspond to classes from the http://dbpedia.org/class/yago/ namespace. Although workers could access the description of these classes via a Web browser, no human-readable information from these URIs could be dereferenced to facilitate the crowd the understanding of the triple object.
- 35 of the false positives in this stage correspond to triples whose objects are external Web pages.
- The predicates of the rest of the misclassified triples correspond to owl:sameAs (in 18 RDF triples), dbpedia-prop:wordnet_type (one RDF triple), and dbpedia-owl:termPeriod (one RDF triple).

In the *Find* stage, the crowd achieved similar values of precision in both settings 'first answer' and 'majority voting'. Furthermore, in this stage the crowd achieved higher precision (0.5291 for 'majority voting') than in the *Find* stage. From the 167 RDF triples with predicate rdf:type, the crowd correctly classified 67 triples. Although the false positives were reduced in the *Verify* stage, the number of misclassified triples with RDF resources as objects are still high. Since the value of inter-rater agreement for this type of task is high, we can deduce that false positives are not necessarily generated by chance but the crowd recurrently confirms that these RDF triples are incorrect. These results suggest that assessing triples with RDF resources as objects without a proper rendering (human-readable information) is challenging for the crowd. Regarding the triples whose objects are external Web pages, in the *Find* stage the crowd correctly classified 35 out of the 36 triples, which is consistent with the behavior observed for this type of triples assessed in the *Verify* stage of the first workflow.

5.4. Evaluation of Baseline Approaches

We took the same set of resources from DBpedia, that were assigned to the LD experts, and performed baseline approaches for each type of quality issue. The results are discussed in this section.

5.4.1. Object Values, Datatypes and Literals

We use the *Test-Driven quality assessment* (TDQA) methodology [21] as our main baseline comparison approach to detect incorrect object values and datatypes and literals. TDQA is inspired from test-driven development and proposes a methodology to define (i) automatic, (ii) semi-automatic and (iii) manual test cases based on SPARQL. Automatic test cases are generated based on schema constraints. The methodology suggests the use of semi-automatic schema enrichment that, in turn, will generate more automatic test cases. Manual test cases are written by domain experts and can be based either on a test case pattern library, or written as manual SPARQL queries.

Table 7
Aggregation of errors based on test case source.

TC Source	TC	Success	Fail	Errors
Automatic	3376	3341	65	424
Enriched	1723	1660	63	137
Manual	47	7	10	204
Total	5146	5008	138	765

RDFUnit[18] [20] is a tool that implements the TDQA methodology. RDFUnit generates automatic test cases for all enabled schemata and checks for common axiom validations. At the time of writing, RDFUnit supports the detection of inconsistencies for *domain* and *range* for RDFS and *cardinality*, *disjointness*, *functionality*, *symetricity* and *reflexiveness* for OWL under Closed World Assumption (CWA). We re-used the same setup for DBpedia, used by Kontokostas et al. [21] but excluding 830 test cases that were automatically generated for rdfs:range. The dataset was checked against the following schemata (namespaces): dbpedia-owl, foaf, dcterms, dc, skos, and geo[19]. In addition, we re-used the axioms produced by the ontology enrichment step for DBpedia, as described by Kontokostas et al. [21].

In total, $5,146$ tests were run, in particular: $3,376$ were automatically generated from the tested vocabularies or ontologies, $1,723$ from the enrichment step and 47 defined manually. From the $5,146$ total test cases only 138 failed and returned a total 765 individual validation errors. Table 7 aggregates the test case results and violation instances based on the generation type. Although the enrichment based test cases were generated automatically, we distinguish them from those automatic test cases that were based on the original schema.

In Table 8, we aggregate the failed test cases and the total instance violations based on the patterns the test cases were based on. Most of the errors originate from ontological constraints such as functionality, datatype and domain violations. Common violation instances of ontological constraints are multiple birth/death dates

[18]`http://rdfunit.aksw.org`
[19]Schema prefixes as used as defined in *Linked Open Vocabularies* (`http://lov.okfn.org`).

and population values, datatype of xsd:integer instead of xsd:nonNegativeInteger and various rdfs:domain violations. In addition to ontological constraints, manual constraints resulted in violation instances such as: *birth date after the death date* (1), *(probably) invalid postal codes* (13), *persons without a birth date* (51), *persons with death date that should also have a birth date* (3), *a resource with coordinates should be a* dbpedia-owl:Place (16) and *a* dbpedia-owl:Place *should have coordinates* (7).

In addition to the violation instances in Table 8, there exists 51 additional violation instances originated by a test case that was written as a manual SPARQL query and checks weather a person's height is between 0.4 and 2.5 meters. In this specific case, the base unit is meters and the values are extracted as centimetre. Thus, although the results look valid to a user, they are actually wrong.

As a baseline approach a complete direct comparison is not possible except for 85 wrong datatypes and 13 failed regular expressions (cf. Table 8). However, even in this case it is not possible to provide a precision since RDFUnit runs through the whole set of resources and possibly catches errors the LD experts didn't catch since it considers the ontological schema. This is because the LD experts performed triple based evaluation using the TripleCheckMate tool, which does not provide schema information directly. Thus, only those experts who are conversant with the schema might be able to identify those errors. Examples of such inconsistencies are datatype detection that is not defined in the ontology e.g. dates vs numbers ("1935"^^xsd:integer) or erroneous language tags. Also, rdfs:domain violations were not reported from the LD experts since for every triple they had to cross-check the ontology definitions for the evaluated property and the rdf:type statements of the resource. Similar combinations apply for all the other patterns types described in Table 8. RDFUnit was running beyond the isolated triple level that the LD experts and crowd were evaluating and was checking various combinations of triples.

However, using RDFUnit the set of incorrect values and datatypes and literals can be extracted and then fed to the LD experts to verify those errors as some of the errors require human judgement in terms of semantically incorrect triples. For example in case of logical inconsistencies, RDFUnit relies only on domain (dataset) experts to define custom rules, for example the human height constraint.

5.4.2. Interlinks

For this type of task, we implemented a baseline that retrieves, for each triple, the external web page – which corresponds to the object of the triple – and searches for occurrences of the foaf:name of the subject within the page. If the number of occurrences is greater than 1, the algorithm interprets the external page as being related to the resource. In this case the link is considered correct. Listing 1 shows the script used to detect the number of times the title of the resource appears in the web page.

Listing 1: Script for detecting whether the interlink is correct where $a is the link to the external page and $b is the title of the resource.

```
while read a b; do
  wget –qO– $a | grep "$b" | wc –l
done < links.txt
```

In order to compare the baseline with the crowdsourcing approaches (i.e. detection whether the interlinks are correct), we first extracted the interlinks from the triples subject to crowdsourcing. A total of 2, 780 interlinks were retrieved. Table 9 shows the number and types of interlinks present in the dataset.

As a result of running this script, we detected a total of 2412 interlinks that were not detected to have the title of the resource in the external web page (link). In other words, only 368 of the total 2, 780 interlinks were detected to be correct by this automatic approach, achieving a precision of 0.1323.

6. Final Discussions

Referring back to the research questions formulated in Section 1, our experiments let us identify the strengths and weaknesses of applying crowdsourcing mechanisms for data quality assessment, following the *Find-Fix-Verify* pattern. Regarding the precision achieved in both workflows, we compared the outcomes produced in each stage by the different crowds against a manually defined gold standard; precision values achieved by both crowds show that crowdsourcing workflows offer feasible solutions to enhance the quality of Linked Data data sets **(RQ1)**.

In each type of task, the LD experts and MTurk workers applied different skills and strategies to solve the assignments successfully **(RQ2)**. The data collected for each type of task suggests that the effort

Table 8

Aggregation of errors based on the source pattern. We provide the pattern, the number of failed test cases for the pattern (F.TCs) along with the total violation instances (Total) and based on the test case generation type: automatic (Aut.), enriched (Ern.) and manual (Man.).

Pattern Type	**F. TCs**	**Total**	**Aut.**	**Enr.**	**Man.**
`Assymetric (owl)`	2	1	-	1	-
`Cardinality (owl)`	65	142	6	136	-
`Disjoint class (owl)`	1	1	1	-	-
`Domain (rdfs)`	33	363	332	-	31
`Datatype (rdfs)`	29	85	85	-	-
`Comparison`	1	1	-	-	1
`Regular expression constraint`	1	13	-	-	13
`Type dependencies`	3	54	-	-	54
`Type-property dependencies`	1	51	-	-	51
`Property dependencies`	1	3	-	-	3
Total	**137**	**714**	**424**	**137**	**153**

Table 9

Number and the types of interlinks present in the dataset verified by the experts in the contest.

Interlink Type	**Instances**	**Correctly detected**
`http://dbpedia.org/ontology/influencedBy`	23	0
`http://dbpedia.org/ontology/thumbnail`	192	10
`http://dbpedia.org/ontology/wikiPageExternalLink`	1209	163
`http://dbpedia.org/property/wikiPageUsesTemplate`	595	63
`http://dbpedia.org/property/wordnet_type`	82	19
`http://www.w3.org/2002/07/owl#sameAs`	392	70
`http://xmlns.com/foaf/0.1/depiction`	192	26
`http://xmlns.com/foaf/0.1/homepage`	95	17
Total	2780	368

of LD experts must be applied on tasks demanding specific-domain skills beyond common knowledge. For instance, LD experts successfully identified issues on very specific datatypes, e.g., when time units are simply annotated as numbers (xsd:Integer or xsd:Float). In the same type of task, workers focused on assessing triples annotated with language tags, instead of datatypes like the experts. The MTurk crowd has proven to be very skilled at verifying whether literals are written in a certain language. In addition, workers were exceptionally good and efficient at performing comparisons between data entries, specially when some contextual information is provided.

Furthermore, we were able to detect common cases in which none of the two forms of crowdsourcing we studied seem to be feasible. The most problematic task for the LD experts was the one about discerning whether a web page is related to an RDF resource. Although the experimental data does not provide insights into this behavior, we are inclined to believe that this is due to the relatively higher effort required by this specific type of task, which involves checking an additional site outside the *TripleCheckMate* tool. Although the crowd outperformed the experts in *finding* incorrect 'interlinks', the MTurk crowd is not sufficiently capable of assessing links that correspond to RDF resources. Furthermore, MTurk workers did not perform so well on tasks about datatypes where they recurrently confused numerical datatypes with time units.

The observed results suggest that LD experts and crowd workers offer complementary strengths that can be exploited not only in different assessment iterations or stages (**RQ3**) but also in particular subspaces of quality issues. LD experts exhibited a good performance when *finding* incorrect object values and datatypes (in particular, numerical datatypes). In turn, microtask crowdsourcing can be effectively applied to: i) *verify* whether objects values are incorrect, ii) *verify* literals annotated with language tags, and iii) *find* and *verify* incorrect links of RDF resources to web pages.

One of the goals of our work is to investigate how the contributions of crowdsourcing approaches can be integrated into LD curation processes, by evaluating the performance of two crowdsourcing workflows in a cost-efficient way. In microtask settings, the first challenge is then to reduce the amount of tasks submitted to the crowd and the number of requested assignments (different answers), since both of these factors determine the overall cost of crowdsourcing projects. For the *Find* stage, Algorithm 1 generated $2,339$ HITs to crowdsource $68,976$ RDF triples, consistently with the property stated by Proposition 2. In our experiments, we approved a total of $2,294$ task solutions in the *Find* stage and, considering the payment per HIT (US\$ 0.06), the total cost of this evaluation resulted in US\$ 137.58. Furthermore, in the *Verify* stage, the cost of submitting to MTurk the problematic triples found by the experts was only US\$ 43.

7. Related Work

Our work is situated at the intersection of the following research areas: *Crowdsourcing Linked Data management* and *Web data quality assessment*.

7.1. Using Crowdsourcing in Linked Data Management

There is wide agreement in the community that specific aspects of Linked Data management are inherently human-driven [2]. This holds true most notably for those Linked Data tasks which require a substantial amount of domain knowledge or detailed, context-specific insight that go beyond the assumptions and natural limitations of algorithmic approaches.

Like any Web-centric community of its kind, Linked Data has had its share of volunteer initiatives, including the Linked Open Data Cloud itself and DBpedia [23], and competitions such as the yearly Semantic Web Challenge[20] and the European Data Innovator Award.[21]

[20]http://challenge.semanticweb.org/
[21]http://2013.data-forum.eu/tags/european-data-innovator-award

From a process point of view, [41] introduced a methodology for publishing Linked Data. They discussed activities which theoretically could be subject to crowdsourcing, but did not discuss such aspects explicitly. Similarly, [25] tried to map ontology engineering methodologies to Linked Data practice, drawing on insights from interviews with practitioners and quantitative analysis. A more focused account of the use of human and crowd intelligence in Linked Data management is offered in [36]. The authors investigated several technically oriented scenarios in order to identify lower-level tasks and analyze the extent to which they can be feasibly automated. In this context, feasibility referred primarily to the trade-off between the effort associated with the usage of a given tool targeting automation - including aspects such as getting familiar with the tool, but more importantly creating training data sets and examples, configuring the tool and validating (intermediary) results - and the quality of the outcomes. The fundamental question the work attempted to answer was related to ours, though not focused on quality assurance and repair – their aim was come up with patterns for human and machine-driven computation, which could service semantic data management scenarios effectively. This was also at the core of [35], which took the main findings of this analysis a step further and proposed a methodology to build incentivized Semantic Web applications, including guidelines for mechanism design which are compatible to our fix-find-verify workflow. They have also analyzed motivators and incentives for several types of Semantic Web tasks, from ontology population to semantic annotation.

An important prerequisite to any participatory exercise is the ability of the crowd – experts of laymen – to engage with the given data management tasks. This has been subject to several user experience design studies [26,30,34,40,39], which informed the implementation of our crowdsourcing projects, both the contest, and the paid microtasks running on Mechanical Turk. For instance, microtasks have been used for entity linking [8] quality assurance, resource management [42] and ontology alignment [33].

At a more technical level, many Linked Data management tasks have already been subject to human computation, be that in the form of games with a purpose [27,38] or, closer to our work, paid microtasks. Games with a purpose, which capitalize on entertainment, intellectual challenge, competition, and reputation, offer another mechanism to engage with a broad user base. In the field of semantic technologies, the

OntoGame series proposes several games that deal with the task of data interlinking, be that in its ontology alignment instance (SpotTheLink [38]) or multimedia interlinking (SeaFish [37]). Similar ideas are implemented in GuessWhat?!, a selection-agreement game which uses URIs from DBpedia, Freebase and OpenCyc as input to the interlinking process [27]. While OntoGame looks into game mechanics and game narratives and their applicability to finding similar entities and other types of correspondences, our research studies an alternative crowdsourcing strategy that is based on financial rewards in a microtask platform. Most relevant for our work are the experiments comparing games with a purpose and paid microtasks, which showed the complementarity of the two forms of crowdsourcing [32,9].

A similar study is discussed in [28] for ontology alignment. McCann and colleagues studied motivators and incentives in ontology alignment. They investigated a combination of volunteer and paid user involvement to validate automatically generated alignments formulated as natural-language questions. While this proposal shares many commonalities with the CrowdMap [33] approach, the evaluation of their solution is based on a much more constrained experiment that did not rely on a real-world labor marketplace and associated work force.

Web data quality assessment. Existing frameworks for the quality assessment of the Web of Data can be broadly classified as automated (e.g. [15]), semi-automated (e.g. [11]) and manual (e.g.[4,29]). In particular, for the quality issues used in our experiments, [15] performs quality assessment on links but it fully automated and thus is limited as it does allow the user to choose the input dataset. Also, the incorrect interlinks detected require human verification as they do not take the semantics into account. On the other hand, for detection of incorrect object values and datatypes and literals, the SWIQA framework [14] can be used by utilizing different outlier and clustering techniques. However, it lacks specific syntactical rules to detect all of the errors and requires knowledge of the underlying schema for the user to specify these rules. Other researchers analyzed the quality of Web [5] and RDF [17] data. The second study focuses on errors occurred during the publication of Linked Data sets. Recently, a study [18] looked into four million RDF/XML documents to analyze Linked Data conformance. These studies performed large-scale quality assessment on LD but are often limited in their ability to produce interpretable results, demand user expertise or are bound to a given data set.

SPARQL Inferencing Notation (SPIN)[22] is a W3C submission aiming at representing rules and constraints on Semantic Web models using SPARQL. The approach described in [13] advocates the use of SPARQL and SPIN for RDF data quality assessment. In a similar way, Fürber et al. [12] define a set of generic SPARQL queries to identify missing or illegal literal values and datatypes and functional dependency violations. Another related approach is the *Pellet Integrity Constraint Validator* ICV[23]. *Pellet ICV* translates OWL integrity constraints into SPARQL queries. A more light-weight RDF constraint syntax, decoupled from SPARQL, is offered from *Shape Expressions* (ShEx) [31] and *IBM Resource Shapes*[24].

8. Conclusions and Future Work

In this paper, we proposed and compared crowdsourcing mechanisms to evaluate the quality of Linked Data (LD); the study was conducted in particular on the DBpedia dataset. Two different types of crowds and mechanisms were investigated for the initial detection of quality issues: object value, datatype and language tags, and interlinks. We secondly focused on adapting the *Find-Fix-Verify* crowdsourcing pattern to exploit the strengths of experts and lay workers and leverage the results from the *Find*-only approaches.

For the first part of our study, the *Find* stage was implemented using a contest-based format to engage with a community of LD experts in discovering and classifying quality issues of DBpedia resources. Contributions obtained through the contest (referring to flawed object values, incorrect datatypes and missing links) and were submitted to Amazon Mechanical Turk (MTurk), where we asked workers to *Verify* them. For the second part, only microtask crowdsourcing was used to perform the *Find* and *Verify* stages on the same set of DBpedia resources used in the first part.

The evaluation of the results showed that it is feasible to crowdsource the detection of flaws in LD sets; in particular, the experiments revealed that (i) lay workers are in fact able to detect certain quality issues with

[22]`http://www.w3.org/Submission/spin-overview/`

[23]`http://clarkparsia.com/pellet/icv/`

[24]`http://www.w3.org/Submission/2014/SUBM-shapes-20140211/`

satisfactory precision; that (ii) experts perform well in identifying triples with 'object value' or 'datatype' issues, and lastly, (iii) the two approaches reveal complementary strengths.

Our methodology is applicable to any LD set and can be easily expanded to cover different types of quality issues. Our findings could also inform the design of the DBpedia extraction tools and related community processes, which already make use of contributions from volunteers to define the underlying mapping rules in different languages. Finally, as with any form of computing, our work will be most useful as part of a broader architecture, in which crowdsourcing is brought together with automatic quality assessment and repair components and integrated into existing data governance frameworks.

Future work will first focus on conducting new experiments to test the value of the crowd for further different types of quality problems as well as for different LD sets from other knowledge domains. In the longer term, we will also concern ourselves with the question of how to optimally integrate crowd contributions – by implementing the *Fix* stage – into curation processes and tools, in particular with respect to the trade-offs of costs and quality between manual and automatic approaches. Another area of research is the integration of baseline approaches before the crowdsourcing step in order to filter out errors that can be detected automatically to further increase the productivity of both LD experts and crowd workers.

Acknowledgements

This work was supported by grants from the European Union's 7th Framework Programme provided for the projects GeoKnow (GA no. 318159), Aligned (GA no. 644055) and LOD2 (GA no. 257943).

References

[1] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer, and J. Lehmann. Crowdsourcing linked data quality assessment. In H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web - ISWC 2013*, volume 8219 of *Lecture Notes in Computer Science*, pages 260–276. Springer Berlin Heidelberg, 2013.

[2] A. Bernstein, J. M. Leimeister, N. Noy, C. Sarasua, and E. Simperl. Crowdsourcing and the semantic web. *Dagstuhl Reports*, 4(7):22–51, 2014.

[3] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.

[4] C. Bizer and R. Cyganiak. Quality-driven information filtering using the WIQA policy framework. *Web Semantics*, 7(1):1 – 10, Jan 2009.

[5] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549, 2008.

[6] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[7] A. P. M. Davis. Tags for identifying languages. http://tools.ietf.org/html/bcp47, September 2009.

[8] G. Demartini, D. Difallah, and P. Cudré-Mauroux. Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *21st International Conference on World Wide Web WWW 2012*, pages 469 – 478. ACM, 2012.

[9] O. Feyisetan, E. Simperl, M. V. Kleek, and N. Shadbolt. Improving paid microtasks through gamification and adaptive furtherance incentives. In *24th International World Wide Web Conference*, 2015. To Appear.

[10] J. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

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

[12] C. Fürber and M. Hepp. Using semantic web resources for data quality management. In P. Cimiano and H. Pinto, editors, *Proceedings of the 17th international conference on Knowledge engineering and management by the masses (EKAW)*, volume 6317 of *Lecture Notes in Computer Science*, pages 211–225. Springer Berlin Heidelberg, 2010.

[13] C. Fürber and M. Hepp. Using SPARQL and SPIN for data quality management on the semantic web. In W. Abramowicz and R. Tolksdorf, editors, *Business Information Systems*, volume 47 of *Lecture Notes in Business Information Processing*, pages 35–46, 2010.

[14] C. Fürber and M. Hepp. SWIQA - a semantic web information quality assessment framework. In V. K. Tuunainen, M. Rossi, and J. Nandhakumar, editors, *Proceedings of the 19th European Conference on Information Systems (ECIS)*, volume 15, pages 19–30. IEEE Computer Society, 2011.

[15] C. Guéret, P. T. Groth, C. Stadler, and J. Lehmann. Assessing linked data mappings using network measures. In *Proceedings of the 9th Extended Semantic Web Conference*, volume 7295 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2012.

[16] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space (1st edition)*, volume 1 of *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan & Claypool, 2011.

[17] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. Weaving the pedantic web. In C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas, editors, *3rd Linked Data on the Web Workshop at WW*, volume 628, Raleigh, North Carolina, USA, 2010. CEUR Workshop Proceedings.

[18] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, and

S. Decker. An empirical survey of linked data conformance. *Journal of Web Semantics*, 14:14–44, July 2012.

[19] J. Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6), 06 2006.

[20] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, and R. Cornelissen. Databugger: A test-driven framework for debugging the web of data. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 115–118, 2014.

[21] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 747–758, 2014.

[22] D. Kontokostas, A. Zaveri, S. Auer, and J. Lehmann. TripleCheckMate: A Tool for Crowdsourcing the Quality Assessment of Linked Data. In *Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web*, 2013.

[23] J. Lehmann, C. Bizer, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[24] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, et al. DBpedia–A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 2014.

[25] M. Luczak-Rösch, E. Simperl, S. Stadtmüller, and T. Käfer. The role of ontology engineering in linked data publishing and management: An empirical study. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(3):74–91, 2014.

[26] m. c. Schraefel and L. Rutledge, editors. *Special Issue User Interaction in Semantic Web Research*, volume 8(4) of *Journal of Web Semantics*, 2010.

[27] T. Markotschi and J. Völker. GuessWhat?! - Human Intelligence for Mining Linked Data. In *Proceedings of the Workshop on Knowledge Injection into and Extraction from Linked Data at EKAW*, 2010.

[28] R. McCann, W. Shen, and A. Doan. Matching schemas in online communities: A web 2.0 approach. In G. Alonso, J. A. Blakeley, and A. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE*, pages 110–119, 2008.

[29] B. C. Mendes P.N., Mühleisen H. Sieve: Linked data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshop*, pages 116–123. ACM, 2012.

[30] I. Popov. mashpoint: Supporting Data-centric Navigation on the Web. In *Proceedings of the 2012 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems CHI2012*, pages 2249–2254, 2012.

[31] E. Prud'hommeaux, J. E. Labra Gayo, and H. Solbrig. Shape Expressions: An RDF Validation and Transformation Language. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 32–40, New York, NY, USA, 2014. ACM.

[32] M. Sabou, K. Bontcheva, A. Scharl, and M. Föls. Games with a purpose or mechanised labour? a comparative study. In S. Lindstaedt and M. Granitzer, editors, *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*. ACM, 2013.

[33] C. Sarasua, E. Simperl, and N. Noy. CrowdMap: Crowdsourcing Ontology Alignment with Microtasks. In *The Semantic Web - ISWC 2012*, Lecture Notes in Computer Science, pages 525–541. Springer Berlin Heidelberg, 2012.

[34] m. Schraefel, J. Golbeck, D. Degler, A. Bernstein, and L. Rutledge. Semantic Web User Interactions: Exploring HCI Challenges. In *Proceedings of the 2008 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems CHI2008*, pages 3929–3932. ACM, 2008.

[35] E. Simperl, R. Cuel, and M. Stein. *Incentive-Centric semantic web application engineering*, volume 4 of *Synthesis lectures on the semantic web, theory and technology*. Morgan & Claypool Publishers, 2013.

[36] K. Siorpaes and E. Simperl. Human intelligence in the process of semantic content creation. *World Wide Web*, 13(1-2):33–59, 2010.

[37] S. Thaler, K. Siorpaes, D. Mear, E. Simperl, and C. Goodman. Seafish: A game for collaborative and visual image annotation and interlinking. In *The Semanic Web: Research and Applications*, volume 6644 of *LNCS*, pages 466–470. Springer Berlin Heidelberg, 2011.

[38] S. Thaler, K. Siorpaes, and E. Simperl. SpotTheLink: A Game for Ontology Alignment. In *Proceedings of the 6th Conference for Professional Knowledge Management*. ACM, 2011.

[39] V. Uren, Y. Lei, V. Lopez, H. Liu, E. Motta, and M. Giordanino. The usability of semantic search tools: A review. *Knowledge Engineering Review*, 22(4):361–377, 2007.

[40] M. Van Kleek, D. A. Smith, H. S. Packer, J. Skinner, and N. R. Shadbolt. Carpé data: supporting serendipitous data integration in personal information management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2339–2348. ACM, 2013.

[41] B. Villazón-Terrazas and O. Corcho. Methodological guidelines for publishing linked data. *Una Profesión, un futuro: actas de las XII Jornadas Españolas de Documentación: Málaga*, 25(26):20, 2011.

[42] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, July 2012.

[43] A. Zaveri, D. Kontokostas, M. A. Sherif, L. Bühmann, M. Morsey, S. Auer, and J. Lehmann. User-driven Quality Evaluation of DBpedia. In M. Sabou, E. Blomqvist, T. D. Noia, H. Sack, and T. Pellegrini, editors, *Proceedings of 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*, pages 97–104. ACM, 2013.

[44] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality Assessment Methodologies for Linked Data: A Survey. *Semantic Web Journal*, 2015. http://www.semantic-web-journal.net/content/quality-assessment-linked-data-survey.