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 study 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 [20]. However, seamless consumption of LD in applications is still very limited given the varying quality of the data published in the Linking Open Data (LOD) Cloud [22,59]. Data quality is commonly conceived as "fitness for use" [24] for a certain application or use case. In particular, data quality issues in LOD are the result of a combination of data- and process-related factors. The datasets being released into the LOD Cloud are – apart from any factual flaws that 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 [13,18,19,32,38,39]. However, certain 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 [18] or test cases [26], 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 [23] 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 in dividing a task into several smaller subtasks that can be independently solved. Depending 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 a previous work of ours [58] we investigated common quality problems encountered in the DBpedia dataset [28]. We analyzed the detected quality issues and classified them according to the extent to which they could be amenable to crowdsourcing. Based on these results, in this work we study the assessment via crowdsourcing of three specific LD quality issues in DBpedia: incorrect objects, incorrect datatype or language tag, and incorrect link. In the following, we explain the research questions investigated in this work and present our proposed approach.

The first research question explored is hence: **RQ1**: *Is it feasible to detect the studied LD quality issues via crowdsourcing mechanisms?* This question aims at providing insights whether crowdsourcing approaches can be applied to find the selected quality issues in LD sets – specifically in DBpedia – 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 the studied LD quality issues and 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 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 also investigate how well LD experts perform in a crowdsourcing setting and if and how they outperform lay users. And lastly, it is of interest whether one of the two distinct crowd (experts vs. lay users) performs well in areas that might not be a strength of the other crowd.

To answer these questions, we (i) first launched a contest that acquired 58 experts knowledgeable in LD to find and classify erroneous RDF triples from DBpedia (Section 4.1). They inspected $33,404$ triples in total. These triples were then (ii) submitted as paid microtasks on the MTurk platform to be examined by laymen or 'workers' in a similar way (Section 4.2). Each approach (contest and paid microtasks) made 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 workers, as discussed in Section 5, indicate that (i) untrained workers are in fact able to spot certain quality issues with satisfactory precision; that (ii) experts perform well detecting 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 (in terms of accuracy) using lay users or experts for detecting LD quality issues, beyond one-step solutions for pointing out quality flaws?* To do so, we adapted the crowdsourcing pattern known as *Find-Fix-Verify*, originally proposed by Bernstein et al. [4]. 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 detecting quality issues we had 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

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

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 showed 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 demonstrated also that (iii) lay users-only *Find-Verify* approaches can be a viable alternative for detection of the studied LD quality issues if experts are not available and that they certainly outperform *Find*-only lay user workflows.

Note that in this work we did not implement a *Fix* step from the *Find-Fix-Verify* pattern, 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 the studied scenario, we further executed the semi-automatic RDFUnit framework [26] and a simple automatic baseline and compared their outcomes to the results of our crowdsourcing experiments. We showed that while these (semi-)automatic approaches may be amenable to pre-filtering RDF 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 [2], 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 is 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 empirical study by executing the state-of-the-art solution RDFUnit [26], 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

Data quality is commonly conceived as "fitness for use" [24] for a certain application or use case. The Web of Data spans a network of data sources of varying quality. There are a large number of high-quality datasets, for instance, in the life-science domain, which are the result of decades of thorough curation and have been recently made available as Linked Data[2]. Other datasets, 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 [28]. 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 LD quality issues focuses on DBpedia due to the diversity of the vast domain and scope of the dataset. In our previous work [59], we

[2]For example, `http://beta.bio2rdf.org/`

surveyed existing literature and identified a total of 18 the data quality dimensions (criteria) applicable to LD quality assessment. We classified these dimensions into four groups: (i) intrinsic, those that are independent of the user's context; (ii) contextual, those that highly depend on the context of the task at hand, (iii) representational, those that capture aspects related to the design of the data and (iv) accessibility, those that involve aspects related to the access, authenticity and retrieval of data to obtain either the entire or some portion of the data (or from another source) for a particular use case. In our previous experiment [58], we identified the quality dimensions applicable to DBpedia and found that four dimensions were found particularly prevalent: *Accuracy*, *Relevancy*, *Representational-Consistency* and *Interlinking*. To provided a comprehensive analysis of DBpedia quality, we further divided these four quality dimensions into 7 categories and 17 sub-categories [58].

For the purpose of this paper, from these subcategories we chose the following three triple-level quality issues belonging to two dimensions: (i) Object incorrectly/incompletely extracted belonging to the Accuracy dimension; (ii) Datatype or language tag incorrectly extracted belonging to the Accuracy dimension; and (iii) Incorrect link belonging to the Interlinking dimension. While (i) and (ii) belong to the intrinsic group, (iii) is part of the accessibility group. These categories of quality problems were specifically chosen because, according to our previous study [58], these were highly frequent occurring problems in DBpedia (version 3.9). The selected quality issues are described with examples below.[3]

Object incorrectly/incompletely extracted. Consider the following triple: (dbpedia:Rodrigo_Salinas, dbo:birthPlace, dbpedia:Puebla_F.C.). The DBpedia resource is about the soccer player '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 or language tag for a typed literal in the object position. For example, consider the triple: (dbpedia:Vicks, rdfs:label, "Vicks"@de). The language tag of the literal is considered incorrect since the correct spelling in German for this company/brand is 'Wick'[4].

Incorrect link. This category refers to RDF triples whose association between the subject and the object is incorrect. This occurs when objects do not show any related content pertaining to the subject of the triple. Erroneous interlinks can associate resources within a dataset or between several data sources. This category of quality issues also includes faulty links to external Web sites or other external data sources such as Wikipedia, Freebase, or GeoSpecies, among others.

Given the diversity of situations in which the selected quality issues can be instantiated (broad range of object values and datatypes) and their semantic character, assessing them automatically is challenging. Current automatic approaches apply different mechanisms to detect various types of errors in LD datasets, for example: inconsistencies with ontological definitions [26,52], assigning missing classes to RDF resources [39], or abnormal numerical values [13,32]. Also semi-automatic solutions [26] have been proposed that rely on domain experts to specify customized rules that are tested against the dataset. Further details about these approaches and other relevant works are presented in Section 7. Although these approaches are able to reliably identify certain issues in LD, there are still a considerable amount of errors that are missed, in particular those related to semantic correctness of facts. We therefore theorize that human-based appraisal can constitute an effective solution to detect the selected quality flaws in many instances. In particular, these three quality issues require different cognitive skills in terms of evaluation, i.e. from examining the values of different attributes to identifying whether the links between two resources is appropriate. In this way, we can study whether it is feasible to evaluate these types of quality issues via crowdsourcing mechanisms where lay users can identify erroneous triples without having knowledge about the underlying RDF structure. This allows us for identifying which type of skills are most cost-effective to be employed with regards to utilizing crowdsourcing.

3. Crowdsourcing Preliminaries

The term crowdsourcing was first proposed by Howe [23] and consists of a problem-solving mecha-

[3] RDF prefixes used in the examples throughout this paper are defined at `http://dbpedia.org/sparql?nsdecl`.

[4] `https://de.wikipedia.org/wiki/Vicks`

nism 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, contest, macrotask, crowdfunding, among others. Each form of crowdsourcing is designed to target particular types of problems and reaching out to different crowds. Crowdsourcing tasks can be executed in a single iteration, where tasks are submitted to the crowd and the outcome is directly considered the solution of the given problem. However, in order to produce reliable results from crowds, the *Find-Fix-Verify* pattern has been proposed [4], in which tasks are carried out in successive verification stages. In the following we briefly describe the two crowdsourcing methods studied in this work – contest-based and microtask crowdsourcing – as well as the *Find-Fix-Verify* pattern.

3.1. Types of Crowdsourcing Employed 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. In a crowdsourcing setting, contests exploit 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) [31,49]. In particular, contests as means to successfully involve experts in advancing science have a long-standing tradition in research, e.g., the Darpa challenges[5] and NetFlix.[6] 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 in the 'DBpedia Evaluation Campaign'[7] 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 evaluated the highest number of DBpedia resources. Further details about this contest are explained in Section 4.1.

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') [23]. Microtask crowdsourcing works best for tasks that rely primarily on basic human abilities, such as audio and visual 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[8]. 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 contributors (or 'workers') allow for techniques such as majority voting (or other aggregation methods) to automatically identify accurate responses. The most common reward model in microtask 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 perform each microtask according to the type of studied 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:[9]

- *Requester:* User that submits tasks to the platform (MTurk).
- *Human Intelligence Task (HIT):* Work unit in MTurk and refers to a single microtask. A HIT is a self-contained task submitted by a requester.
- *Worker:* Human contributor 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 re-

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

[6] http://www.netflixprize.com/

[7] http://nl.dbpedia.org:8080/TripleCheckMate/

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

[9] http://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/mechanical-turk-concepts.html

fer 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 provides a fixed set of qualification types, including "Approval Rate" defined as the percentage of tasks successfully solved by a worker. Requesters can also create customized qualification types.

3.2. Crowdsourcing Pattern Find-Fix-Verify

The *Find-Fix-Verify* pattern [4] consists in 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. Lastly, the *Verify* stage corresponds to a final quality control iteration.

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 [4]. 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 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 crowds, as they require different skills and expertise [4].

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 a set $\mathcal{T}$ of RDF triples and a set $\mathcal{Q}$ 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' (with respect to Q), otherwise it can be affirmed that t is 'correct' (with respect to Q).*

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* [4]. 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 adaptation of the *Find-Fix-Verify* pattern consists in executing only the *Find* and *Verify* stages. 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. Furthermore, our adaptation of the *Find-Fix-Verify* pattern is tailored to assess the quality of LD datasets that are (semi-)automatically created from other sources. Such is the case of DBpedia [30], a dataset created by extracting knowledge from Wikipedia via declarative wrappers or mappings. The DBpedia wrappers are the result of a crowdsourced community effort of contributors to the DBpedia project. When datasets are extracted via wrappers or mappings, it is highly probable that the quality issues detected for a certain triple might also occur in the set of triples that were generated with 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.

We devise a two-fold approach to crowdsource triple-based quality assessment of (semi-)automatically

extracted LD datasets. Our approach relies on the *Find* and *Verify* stages of the *Find-Fix-Verify* crowdsourcing pattern. In the *Find* stage, the crowd is 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 in 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 $\dot{\phi}(\mathcal{T})$, the Verify stage consists in crowdsourcing mappings as follows $\ddot{\phi} : \dot{\phi}(\mathcal{T}) \rightarrow 2^{\dot{\phi}(\mathcal{T})}$. 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 $\mathcal{V}_o = (\mathcal{T}, \ddot{\phi}(\mathcal{T}))$.*

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 experts and lay users. This first workflow leverages the expertise of LD experts in the *Find* stage, carried out as a contest, to find and classify erroneous triples according to a predefined quality taxonomy; workers from a microtask platform then assess the outcome of the contest in *Verify* stage. The second workflow entirely relies on microtask crowdsourcing to perform both the *Find* and the *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 crowds 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 LD enthusiasts via a contest. The tasks in the contest consisted of identifying and classifying specific types of LD quality problems in DBpedia triples. To collect the contributions from this crowd, in previous work [58], we developed a Web-based tool called *TripleCheckMate*[10] [27] (cf. Figure 2). *TripleCheckMate* allowed human contributors to select RDF resources, identify issues related to RDF triples of the resources and classify these issues according to a pre-defined taxonomy of data quality problems. In this work, we configured *TripleCheckMate* to use the DBpedia dataset (version 3.9) and the taxonomy of quality issues presented in Zaveri et al. [58]. However, *TripleCheckMate* can be easily configured to work with any dataset. In the contest, a prize was announced for the user submitting the highest number of (real) quality problems.

The *Find* stage 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 the dataset: (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. In our implementation, 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 [58]) to erroneous 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

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

Table 1
Comparison between the proposed crowdsourcing mechanisms to perform 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)

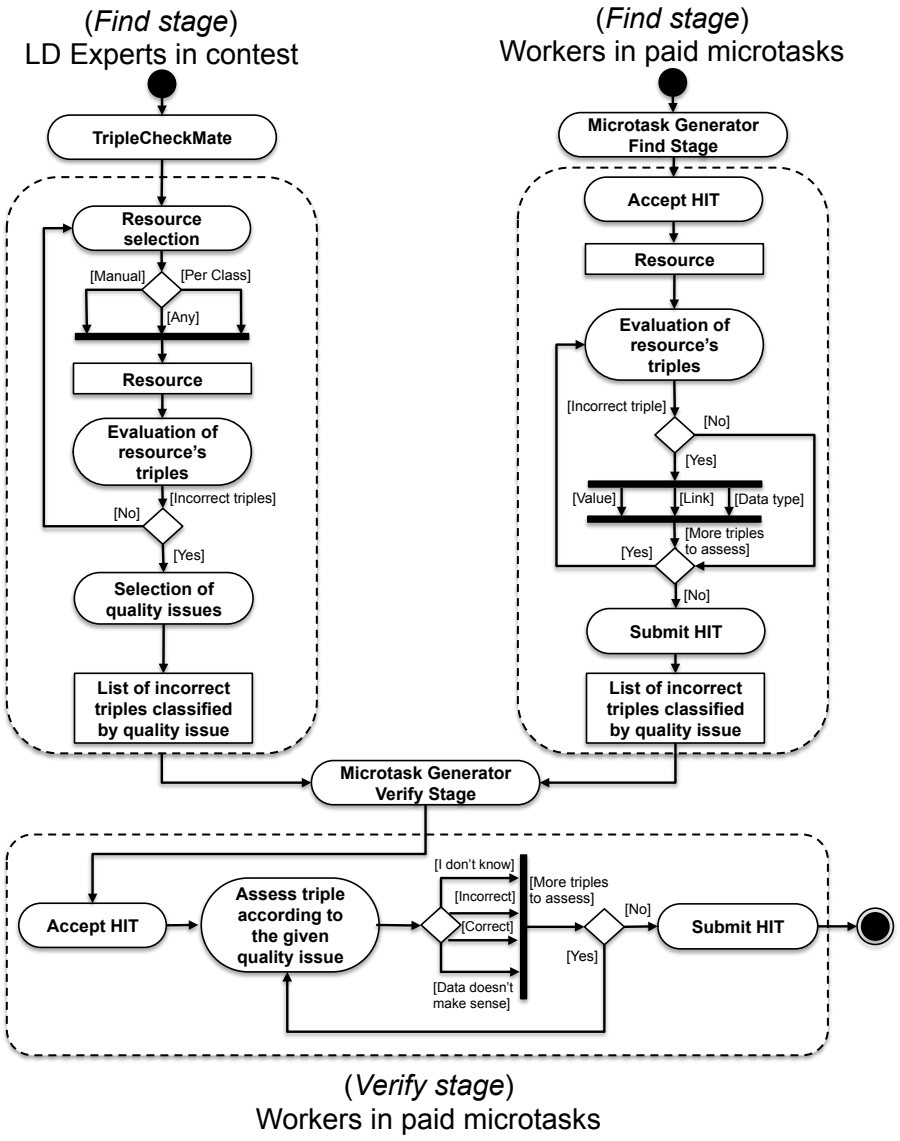


Fig. 1. Studied workflows to crowdsource LD quality assessment. The first workflow combines LD experts reached via a contest with laymen from microtask crowdsourcing. The second workflow solely relies on microtask crowdsourcing.

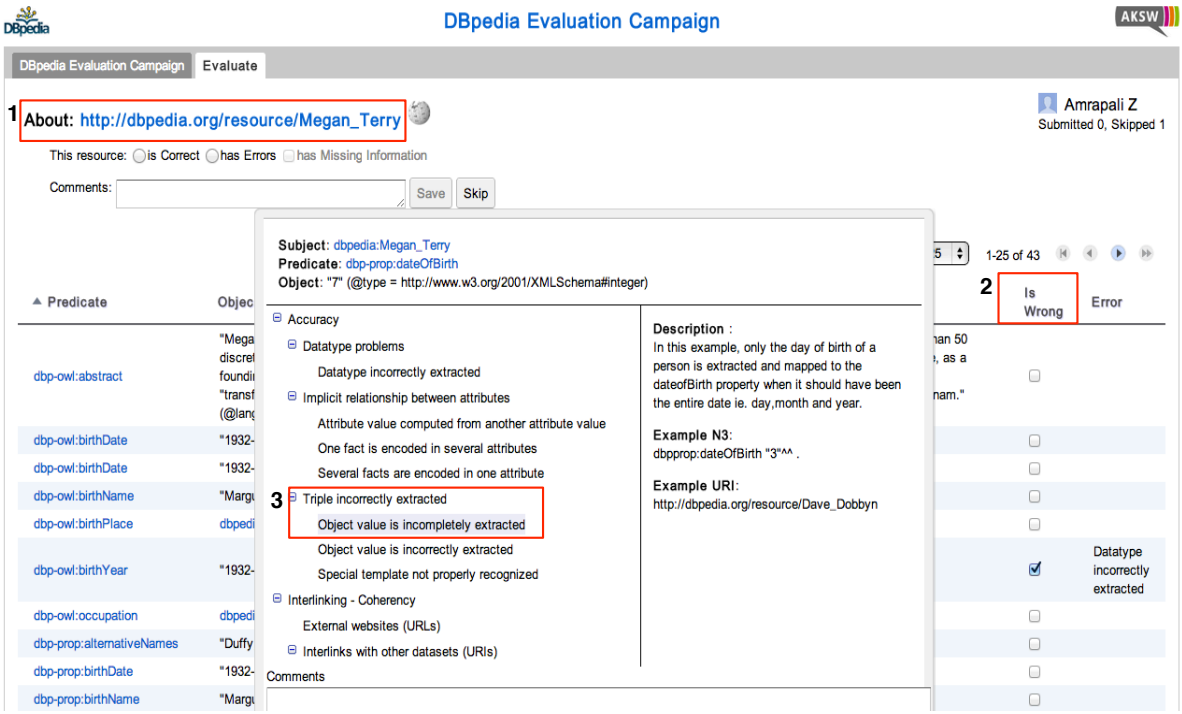


Fig. 2. Screenshot of the *TripleCheckMate* crowdsourcing data quality assessment tool. (1) Displays the RDF resource that is currently being assessed; (2) Users can specify that a triple is erroneous by checking the box 'Is Wrong'; (3) Users select the quality issues present in the triple from a pre-defined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

the definition of the *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 RDF resources are typically checked multiple times. This allows us to (i) analyze the performance of the users (as compared with each other), (ii) to detect unwanted behaviour (as users are not 'rewarded' unless their assessments are 'consensual') and (iii) to ensure the quality of the assessment (i.e. when there is an agreement and several workers detect the same quality issue). 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 that are solved by lay users from a crowdsourcing platform. In this variant of the *Find* stage we aimed at implementing a similar workflow for the crowd workers as the one provided to the LD experts. However, given that crowd workers are not necessarily knowledgeable about RDF or complex taxonomies of LD issues [44], we restricted the scope of LD quality assessment to the issues presented in Section 2. In addition, following the guidelines presented by Sarasua et al. [44], each microtask was augmented with human-readable information that could be dereferenced from RDF triples. Formally, in our approach, a microtask is defined as follows.

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

Following the MTurk terminology (cf. Section 3), each 3-tuple $(t, h_t, \mathcal{Q})$ 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

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 $\mathcal{T}'' \subseteq \mathcal{T}'$ such that $\mathcal{T}'' = \{t | t = (s,p,o) \wedge t \in \mathcal{T}'\}$
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** $\mathcal{M}$

to create the microtasks. The algorithm firstly performs a pruning step (line 2) to remove triples that do not require human assessment. The pruning function in our approach is generic and can be implemented differently according to specific use cases. For instance, in our experiments, the function $prune$ simply discards RDF triples whose URIs could not be dereferenced. After the pruning step, the remaining triples are stored in $\mathcal{T}'$. 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 construction of very long tasks, i.e., when the number of triples with the same subject is large. Appropriate values of α enable the creation of tasks that can still be solved in a reasonable time, consistent with the concept of microtask (a short task). The algorithm continues creating microtasks for all the triples of a resource (lines 7-16), for all the resources (lines 4-18). The output of Algorithm 1 is a set $\mathcal{M}$ of microtasks to assess the quality of 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 3. For each triple, the worker determines whether the triple is 'incorrect' with respect to the fixed set of quality issues $\mathcal{Q}$. In our implementation, $\mathcal{Q}$ was composed of the following LD quality issues (cf. Section 2): object incorrectly extracted, datatype-/language tag incorrectly extracted, or incorrect link, abbreviated as 'Value', 'Datatype', and 'Link', respectively. The crowd has the possibility to select one or several quality issues per triple. Once the worker has assessed all the triples within a microtask, she proceeds to submit the HIT. Consistently with the *Find* stage implemented with the 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 starting 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 RDF 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 the conversion from

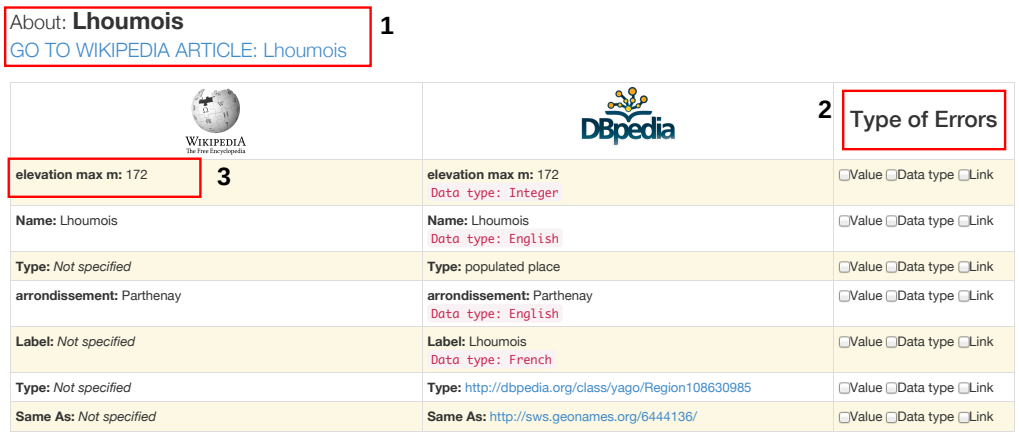


Fig. 3. Screenshot of a microtask generated in the *Find* stage. (1) Displays the RDF resource that is currently assessed and also a link to the Wikipedia page of the resource; (2) Users select the corresponding quality issues present in the triple; (3) Displays contextual information: In our implementation, we extracted values from the infobox of the Wikipedia article associated with the resource – not all the properties of DBpedia resources are available in the infobox, in this case the microtask interface displays *'Not specified'* in the Wikipedia column.

the best current practices BCP 47 [9], as suggested by the RDF specification[11]. Language tags and datatypes of objects were highlighted, such that workers can easily identify them[12]. In addition, contextual information can be displayed in the microtasks in order to assists the workers to solve in successfully solving the task.

Depending on the human-readable data available in the dataset, line 9 from Algorithm 1 could be implemented differently in order to provide contextual information. For constructing our microtasks, we implemented a simple wrapper which extracts data encoded in the infobox of the Wikipedia article version from which DBpedia triples were generated (depicted in the first column of Figure 3). To do so, we crawled the Wikipedia page for the version specified via the property prov:wasDerivedFrom. In the instructions of the microtasks we explained workers that the Wikipedia column should be considered as a guideline; we clarified that these values are not strictly correct[13] or sometimes are not even available[14]. Therefore, to make a better judgement (in case that data in the Wikipedia column is not understandable or not available) workers could visit the corresponding Wikipedia page version by clicking on the provided link in the microtask.

Further microtask design criteria related to quality control is presented in the experimental settings (cf. Section 5.2.2). We used different mechanisms to discourage low-effort behavior which leads to random answers and to identify accurate answers.

The outcome of this stage corresponds to a set of triples $\mathcal{T}$ judged as 'incorrect' by lay users and annotated it with the detected quality issues in $\mathcal{Q}$.

4.3. Verify Stage: Paid Microtask Crowdsourcing

In this stage, we applied microtask crowdsourcing in order to verify quality issues in RDF triples identified as problematic during the *Find Stage* (see Figure 1). To ensure that in this stage a proper validation is performed on each triple, the microtasks are simplified with respect to the ones from the *Find* stage such that: (i) each microtask focuses on a specific quality issue, and (ii) the number of triples per microtask is reduced.

The generation of microtasks in this stage is presented in Algorithm 2. This algorithm groups triples in $\mathcal{T}$ obtained from the previous stage by quality issue, which enables workers to focus on one quality issue at a 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.

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

[12] Datatype and language tag errors do not occur simultaneously in an RDF triple and both are associated with literals in the object position. Therefore, to simplify the instructions, datatypes and language tags are introduced as a single issue to workers, therefore our interfaces display "datatype" even for language tags.

[13] The implemented wrapper could introduce certain errors while parsing the Wikipedia articles' infoboxes.

[14] Certain predicates in DBpedia triples cannot be found in Wikipedia infoboxes, in particular predicates of RDF/S or OWL, e.g., rdfs:label, rdf:type and owl:sameAs. Therefore, the Wikipedia column is usually less complete than the DBpedia one.

Algorithm 2 Microtask Generator for Verify Stage

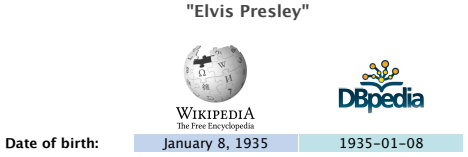
Require: $\mathcal{F}_o = (\mathcal{T}, \dot{\phi}(.))$ and β, where $\mathcal{T}$ is a set of RDF triples, $\dot{\phi}(.)$ is a mapping of triples in $\mathcal{T}$ to quality issues, and β is the maximum number of triples grouped in a single microtask.
Ensure: A set of microtasks $\mathcal{M}$ to assess 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** $(t, \dot{\phi}(.)) \in \mathcal{F}'_o$ **do**
5: **for all** $q \in \dot{\phi}(t)$ **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** $\mathcal{M}$

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* pattern [4]. Further implementations of the $prune$ function could consider agreement or confidence values obtained in the *Find* stage in order to crowdsource a triple in the *Verify* stage[15]. In our second workflow, the function $prune$ discards answers whose inter-rater agreement values were not higher than a certain threshold. 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 3-8). The set $\mathcal{Q}$ contains all the quality issues present in the set of triples $\mathcal{T}$ (line 9). For each quality issue q (line 10), the algorithm processes all triples associated with that quality issue (line 12). The algorithm extracts human-readable information about the triples (line 13) and appends it to the microtask (line 14). In case 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 $\dot{\phi}(.)$ identified in the *Find* stage.

[15] For instance, a low agreement value might suggest that the triple has no quality issues and hence it should not be crowdsourced. On the other hand, a high agreement value could be an indicator that the triple is indeed incorrect and no further verification is needed. Setting appropriate thresholds for agreement in $prune$ might also depend on the expertise of the crowd. However, exploring optimal configurations of the $prune$ function is out of the scope of this work.

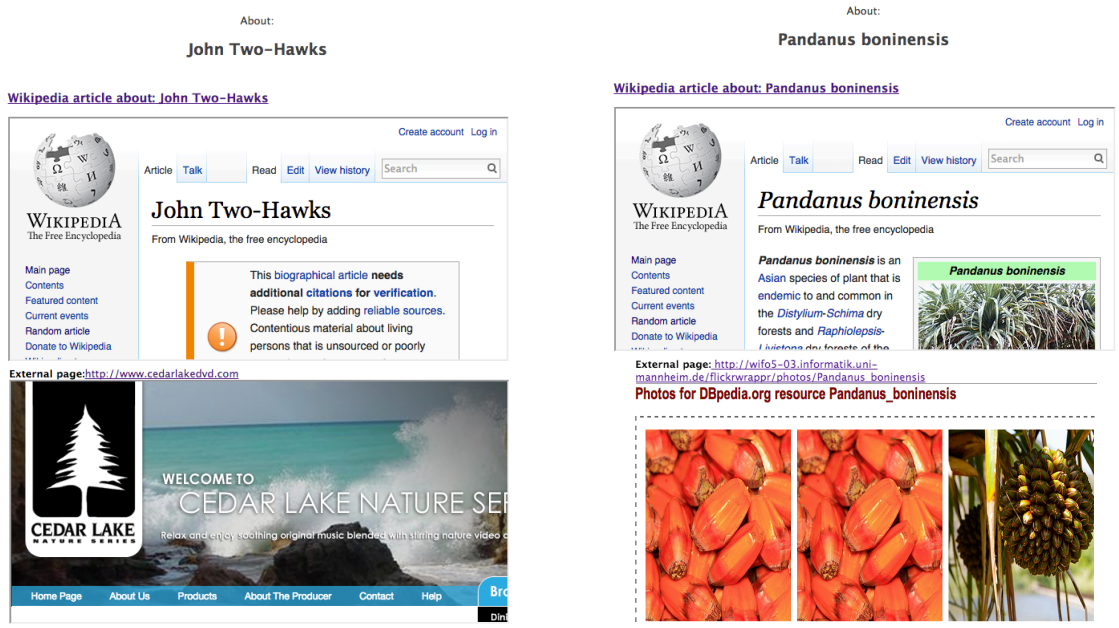
(a) **Incorrect** object value in DBpedia



(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.

Based on the classification of LD quality issues explained in Section 2, Algorithm 2 creates 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 workers 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 users to specify when they could not provide a reliable answer. The fourth option referred to those cases in which the presented data was truly unintelligible. Furthermore, 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.



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

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

Fig. 5. Incorrect link: The crowd must decide whether the content of a link (indicated as "External page" in the user interface) is related to the subject. When assessing links between RDF resources, the preview of the "External page" displays the resource's page (most of the datasets linked from DBpedia – Wikidata, Yago – support Linked Data browsers).

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 URIs of the subject and predicate of triples. 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. n 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.

Incorrect datatypes or literals. 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 concept of data type of a value and provided two simple examples to the crowd. 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 object of an RDF triple is associated with the subject. Incorrect subject-object associations may be due to several reasons: erroneous links referenced from Wikipedia articles, wrong associations between RDF resources (e.g., via the dbp:wordnet_type predicate) within DBpedia, or incorrect extraction of links via the DBpedia wrappers. In the latter case, wrongly extracted links that result in broken links can be automatically detected and are not crowdsourced. For the interface of the HITs, we provided workers a preview of the Wikipedia article and the triple object by implementing HTML iframe tags. In addition, we retrieved the foaf:name of the subject and the link to the corresponding Wikipedia article using the predicate foaf:isPrimaryTopicOf.

Examples of this type of task are depicted in Figure 5. In the first example (see Figure 5a), 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 right 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 directly impacts the scalability of the approach in terms of the time required to solve all the tasks and the overall monetary cost. The following proposition states the complexity 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 in theory 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 ob-

served 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. The main goal of our experiments is studying the applicability of crowdsourcing as a solution to the problem of detecting quality issues in LD datasets. Specifically, we formulated the following research questions:

RQ1: *Is it feasible to detect the studied LD quality issues via crowdsourcing mechanisms?*

RQ2: *In a crowdsourcing approach, can we employ unskilled lay users to identify the studied LD quality issues and to what extent is expert validation needed and desirable?*

RQ3: *How can we design better crowdsourcing workflows (in terms of accuracy) using lay users or experts for detecting LD quality issues, beyond one-step solutions for pointing out quality flaws?*

In addition, we executed (semi-)automatic approaches to detect quality issues which allowed us to understand the strengths and limitations of applying crowdsourcing in this scenario. We used the semi-automatic RDFUnit tool [26] for assessing 'object value' and 'datatype' issues, and implemented a simple automatic 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).[16] 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 in Python 2.7.2. During the experiments, we processed a total of $38,633$ RDF triples from DBpedia and pruned $5,230$ triples that contained non-dereferenceable URIs. A URI was considered non-dereferenceable if after ten retries of performing HTTP GET operations, we did not obtain a successful response (HTTP codes 2xx or 3xx in the response header). Non-dereferenceable URIs corresponded to external datasets or web pages, i.e., DBpedia URIs were dereferenced successfully. In our experiments, we aim at gaining insights on the type of misclassifications performed by experts and laymen, therefore, besides pruning broken links no further RDF triples were removed from our study. With Algorithms 1 and 2, we generated the corresponding microtasks for the *Find* and *Verify* stages, respectively. Resulting microtasks were submitted as HITs to Amazon Mechanical Turk using the MTurk SDK for Java.[17]

5.1.2. Metrics

The task in 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 [14] metric to measure the consensus degree among raters (experts or MTurk workers);
ii) *Precision* to measure the proportions of positive results of each crowd, computed as $\frac{TP}{TP+FP}$.
iii) *Sensitivity* to measure the true positive rate, computed as $\frac{TP}{TP+FN}$
iv) *Specificity* to measure the true negative ratio, computed as $\frac{TN}{TN+FP}$

[16] http://wiki.dbpedia.org/Downloads39

[17] 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* stages, 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	$\mathcal{O}(\|\mathcal{T}\|)$	$\mathcal{O}(\|\mathcal{T}\|\|\mathcal{Q}\|)$
Total tasks generated (only for microtask crowdsourcing)	$\sum_{s_i \in \mathcal{S}} \left\lceil \frac{\|\{(p,o)\|(s_i,p,o)\in\mathcal{T}\}\|}{\alpha} \right\rceil$	$\sum_{q_i \in \mathcal{Q}} \left\lceil \frac{\|\{t\|t\in\mathcal{T} \wedge q_i \in \dot{\phi}(t)\}\|}{\beta} \right\rceil$
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

Inter-rater agreement of the experts is reported by *TripleCheckMate* for the overall results of the contest in the *Find* stage. Therefore, we also report on the inter-rater agreement of the overall results for microtask workers in the *Find* stage. Inter-rater agreement is computed per quality issue for the *Verify* stages.

Precision values were computed for all stages of the studied workflows. In our *Find* stages, the crowd – expert or layman – was not enquired for annotating triples as 'correct' (in conformance with the definition of the *Find-Fix-Verify* pattern [4]); i.e., the outcome of our *Find* stages does not contain true or false negatives. Therefore, sensitivity and specificity values were computed only for the *Verify* stages.

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 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 had 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 [8], 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. Given the effort and careful process (resolution of conflicts and discussion of disputed triples) carried out during our assessment, we consider that the produced gold standard is sufficiently reliable to evaluate the outcome of the different crowdsourcing approaches. We are however making the gold standard available and encourage the community to assess and expand it further.

The tools used in our experiments and the results are available online, including the outcome of the contest,[18] the gold standard and microtask data (HITs and results).[19]

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

5.2.1. Contest Settings: Find Stage

The contest was open from November to December in 2012, and was configured as follows.

[18] http://nl.dbpedia.org:8080/TripleCheckMate/

[19] http://people.aifb.kit.edu/mac/DBpediaQualityAssessment/

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.
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

The microtasks for this experiment were submited to MTurk in May 2013 using the following settings.
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 workers to solve five different questions ($\beta = 5$). Each question corresponded to an RDF triple and each HIT contained 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. At the time of submitting the tasks, 0.04 was one of the most popular HIT rewards in MTurk as reported by Difallah et al. [11].
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 collecting more than our answer (assignment) 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 we determined that the experts browsed around 33,404. 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 and triples whose objects were broken links. In total, we submitted $1,073$ triples to the crowd. A total of 80 distinct workers assessed all the RDF triples in four days. The average time per microtask spent by the crowd is 94.55 sec. for incorrect values, 71.69 sec. for incorrect datatypes or language tags, and 116.11 sec. for incorrect links. We then computed the effective hourly rate per type of task: 1.52 US$ for incorrect values, 2.01 US$ for incorrect datatypes or language tags, and 1.24 US$ for incorrect links. 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 inter-rater agreement as well as precision, sensitivity, and specificity 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 *TripleCheckMate* calculated 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. 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 LD 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 Values

As reported in Table 4, our crowdsourcing experiments reached a precision of 0.8977 for MTurk workers (majority voting) and 0.7151 for LD experts. Most of the incorrect values that are extracted from Wikipedia occur with the predicates related to dates, for example: (dbpedia:2005_Six_Nations_Championship, dbp: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 semantically erroneous, although they were syntactically correctly extracted from Wikipedia. One example of these triples is: (dbpedia:Oncorhynchus, dbp:subdivision, "See text")[20]. We found out that the LD experts classified all these

[20]Wikipedia page version from which this data was extracted: `https://en.wikipedia.org/w/index.php?title=Oncorhynchus&oldid=551701016`

Table 3
Overall results in each type of crowdsourcing approach in first crowdsourcing workflow: Combining LD experts (*Find* stage) and microtask workers (*Verify* stage).

	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 triples	Browsed: 33,404 Marked as 'incorrect': 1,512	Evaluated: 1,073
Object values	550	509
Datatype/Language tags	363	341
Interlinks	599	223

Table 4
Inter-rater agreement and metrics (computed against the Gold Standard) achieved in the first crowdsourcing workflow: Combining LD experts (*Find* stage) and microtask workers (*Verify* stage).

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
		Precision	
Find: LD experts	0.7151	0.8270	0.1525
Verify: MTurk workers (first answer)	0.8595	0.8889	0.6111
Verify: MTurk workers (majority voting)	0.8977	0.9116	0.7674
		Sensitivity	
Verify: MTurk workers (first answer)	0.8056	0.5161	0.8800
Verify: MTurk workers (majority voting)	0.8899	0.4802	0.9705
		Specificity	
Verify: MTurk workers (first answer)	0.6693	0.6897	0.8947
Verify: MTurk workers (majority voting)	0.7482	0.7759	0.9450

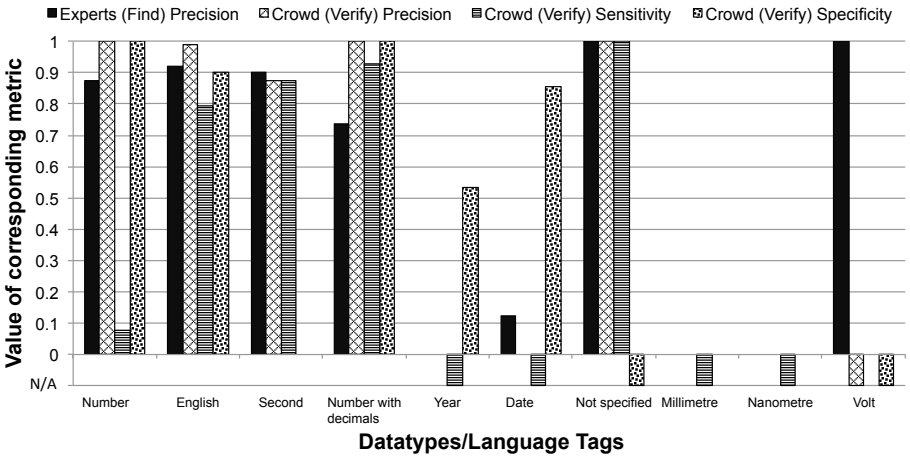
triples as incorrect. In contrast, the workers answered that 50 out of this 52 were correct, since they could easily compare the DBpedia and Wikipedia values in the HITs. Although these observations may impact the specificity reached by crowd workers, we obtained higher values of sensitivity than specificity (0.8899 vs. 0.7482 in majority voting) in both microtask settings. This suggests that workers perform better when detecting incorrect values (true positives) than correct values (true negatives) in RDF triples.

5.2.5. Results: Incorrect Datatypes or Language Tags

Table 4 exhibits that both crowdsourcing mechanisms achieved high values of precision: 0.8270 precision for experts on *finding* this type of quality issue, while the crowd achieved 0.9116 precision on *verifying* these triples. However, a closer inspection to the results revealed that the crowd generated a large number of false negatives, obtaining low sensitivity values (0.4802 with majority voting). In particular, the first answers submitted by the crowd were slightly better in terms of sensitivity than the results obtained with majority voting. A detailed study of these cases showed that 28 triples that were classified correctly in the first answer from the crowd, later were misclassified, and most of these triples refer to a language datatype. The low performance of the MTurk workers in terms of sensitivity is not surprising, since this particular task

(a) Frequency of datatypes and language tags in the crowdsourced triples in the first crowdsourcing workflow.

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) Metrics precision, sensitivity, and specificity per datatype in each stage (*Find, Verify*) in the first crowdsourcing workflow. Bars with values N/A indicate that the metric could not be computed since the denominator was equal to zero.

Fig. 6. Results for the "Incorrect datatype/language tag" task in the first crowdsourcing workflow (combining experts and crowd workers).

requires certain technical knowledge about datatypes and their specification in RDF.

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 and language tags in the assessed triples (see Figure 6a) and reported on precision, sensitivity, and specificity achieved by the crowdsourcing methods per datatype or language tag. 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"[21], for instance, in the DBpedia triple (dbpedia:Walter_Flores, dbp:dateOfBirth, "1933") the value "1933" was typed as number instead of year. These are the cases where the crowd was confused and determined that the datatype 'number' was correct, thus generating a large number of false negatives, hence the low values of sensitivity for this datatype. 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 sensitivity of the crowd would have been 0.85 and 0.82 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. Although the precision achieved by the crowd in this language tag is high, we identified that the crowd is less successful in the following two situations: (i) The value corresponds to a number and the remaining data was specified in English, e.g., (dbpedia:Middelburg, dbo:utcOffset, '+1"@en). (ii) The value is a text without special characters, but in a different language than English – for example German – as in the following triple (dbpedia:Woellersdorf-Steinabrueckl, dbp:art, "Marktgemeinde"@en). The performance of both crowdsourcing approaches for the remaining datatypes were similar or not relevant due the low number of triples processed.

[21]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.

5.2.6. Results: Incorrect Links

Table 4 displays the precision for each studied quality assessment mechanism. The extremely low precision of 0.1525 of the contest's participants was unexpected. We inspected in detail the 189 misclassifications of the experts:

- The 95 Freebase links[22] 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 Wikimedia uploads (composed mostly by images hosted for Wikipedia); 74 of these triples were also classified incorrectly.
- 20 links (to 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.

On the other hand, MTurk workers achieved high values in both setting, in particular when applying majority voting: 0.7674 for precision, 0.9705 for sensitivity, and 0.9450 for specificity as shown in Table 4. 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 with the subject is not straightforward. Examples of these cases are the following subjects and links: the resource dbpedia:Frank_Stanford with the Web site `http://nw-ar.com/drakefield` and the resource dbpedia:Forever_Green `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 submitted to MTurk in February 2014 and 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 two workers in the *Find* stage were candidates for crowdsourcing in the *Verify* stage. The microtasks generated in the subsequent stage were crowdsourced in February 2014 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 33,404 RDF triples and the crowd processed 30,658 triples in 14 days. The microtasks from the *Find* stage were resolved by 187 distinct workers in 83.29 secs. on average at an hourly rate of 2.59 US$. In total, 26,835 triples were identified as erroneous, and classified into the three quality issues studied in this work. Then, we selected random samples from triples identified as erroneous in the *Find* stage from the crowd using majority voting. For sampling, we used the same distribution obtained from the first experiment, i.e., each sample contains the exact same number of triples that were crowdsourced in the *Verify* stage in the first workflow. This allowed us to compare the outcome of the *Verify* stage from both workflows. We crowdsourced then 509 triples for the task of incorrect values, 341 for incorrect datatype or language tag, and 223 for incorrect interlinks. All triples crowdsourced in the *Verify Stage* were assessed by 141 distinct workers in seven days. On average, workers spent 95.59 sec. on resolving a microtask for detecting incorrect values, 53.05 sec. on a microtask for incorrect datatypes or language tags, and 131.48 sec. on a microtask for assessing incorrect links. The effective hourly rates in each type of task were: 1.51 US$ for assessing values, 2.71 US$ for assessing datatypes or language tags, and 1.10 US$ for assessing links. For the incorrect value and incorrect datatype or language tag tasks, all submitted microtasks were finished in the first two days. Regarding the incorrect link tasks, 86% of the microtasks were resolved within four days (consistently with the behavior observed in the first experiment), and the remaining 14% of these tasks were completed after seven days of the beginning of the experiment. A summary of these results and further details are presented in Table 5.

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

Table 5
Overall results in second crowdsourcing workflow: Applying microtask workers in both stages *Find* and *Verify*.

	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	30,658	1,073
Total no. of microtasks generated	2,339	216
Total time	14 days	7 days
Total triples	Browsed: 33,404 Marked as 'incorrect': 26,835	Evaluated: 1,073
Object values	8,691	509
Datatypes/Language tags	13,194	341
Interlinks	13,732	223

Table 6
Inter-rater agreement and metrics (computed against the Gold Standard) achieved in the second crowdsourcing workflow: Applying microtask workers in both stages *Find* and *Verify*.

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
		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
		Sensitivity	
Verify: MTurk workers (first answer)	0.4549	0.7714	0.8478
Verify: MTurk workers (majority voting)	0.9615	0.9111	1.0000
		Specificity	
Verify: MTurk workers (first answer)	0.5432	0.9223	0.4967
Verify: MTurk workers (majority voting)	0.4371	0.9793	0.3916

Similar to the first 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 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 suggested that 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 as well as sensitivity and specificity values for the *Verify* 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 Values

In the *Find* stage, the crowd achieved a precision of 0.3713 for identifying 'incorrect 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 dbp:placeOfBirth, and dbp: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 dbo:wikiPageID and dbo: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[23], 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[24] 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".

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. The crowd generated a large number of false positives (170 in total), therefore, the values of sensitive achieved in both settings were not high. Moreover, for the setting 'majority voting', the value of sensitivity was 0.9615. Errors from the first iteration were reduced in the *Verify* stage, especially in triples with predicates dbp:dateOfBirth and dbp: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, from the analyzed sample of triples we observed that 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 HIT interface: Microtasks were properly displaying non UTF-8 characters used in several languages in DBpedia, e.g., Russian, Japanese, Chinese, among others.

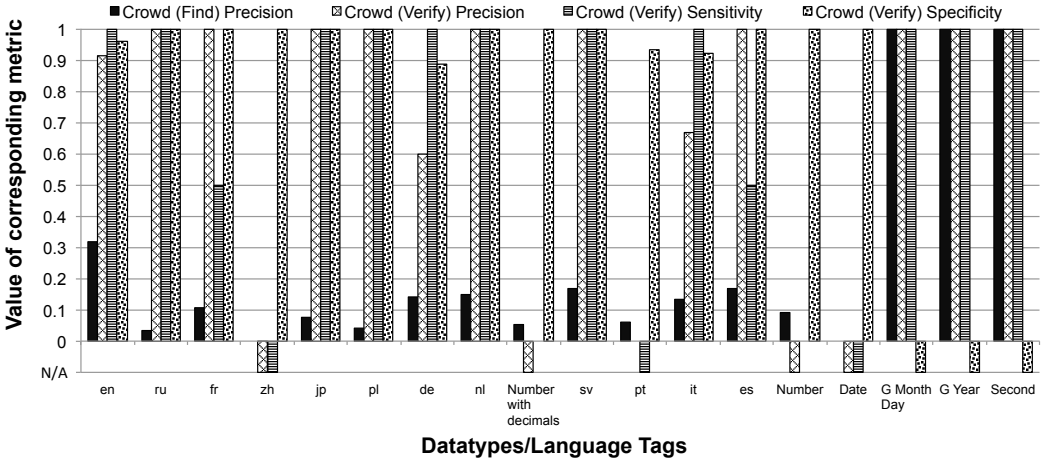
In the *Verify* stage of this type of task, the crowd outperformed the precision of the *Find stage*, achieving values of 0.5510 for the 'first answer' setting and 0.8723 with 'majority voting'. This major improvement on the precision put in evidence the importance of having the a multi-validation pattern like *Find-Fix-Verify* in which initial errors can be reduced in subsequent iterations. For the 'majority voting' setting, the crowd achieved high values for sensitivity (0.9111) and specificity (0.9793) by correctly detecting true positives and true negatives. Congruent with the behavior observed in the first workflow, MTurk workers performed 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 per datatype-/language tag the values for precision for both stages as well as sensitivity and specificity values for the 'majority voting' setting. We can observe that the crowd is exceptionally successful in identifying correct triples in the *Verify* stage that were classified as erroneous in the previous stage (true negatives). This can be confirmed by the high values of sensitivity achieved by the crowd among all the analyzed datatypes/language tags. A

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

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

(a) Frequency of datatypes and language tags in the crowdsourced triples in the second crowdsourcing workflow.

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) Metrics precision, sensitivity, and specificity per datatype in each stage (*Find, Verify*) in the second crowdsourcing workflow. Bars with values N/A indicate that the metric could not be computed since the denominator was equal to zero.

Fig. 7. Results for the "Incorrect datatype/language tag" task in the second crowdsourcing workflow (crowd workers in both stages).

closer inspection to the six false positives revealed that in three cases the crowd misclassified triples whose object is a proper noun with no translation into other languages, for instance, (dbpedia:Tiszaszentimre, foaf:name, "Tiszaszentimre"@en) and (dbpedia:Ferrari_Mythos, rdfs: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, (dbpedia:Book, rdfs:label, "Libro"@it).

5.3.5. Results: Incorrect Links

From the studied sample, the majority of the triples classified as 'incorrect interlink' in the *Find* stage contained objects that correspond to RDF resources. 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 objects of these triples correspond to classes from the http://dbpedia.org/class/yago/ namespace. Workers could not successfully assess these RDF triples, although YAGO URIs in DBpedia are intelligible to some extent[25] and workers could access the description of these URIs via a Web browser. Since no human-readable information is displayed for these URIs, we presume that this might have affected the crowd performance.
- 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), dbp:wordnet_type (one RDF triple), and dbo:termPeriod (one RDF triple).

In the *Find* stage, the crowd achieved similar values of precision in both settings 'first answer' and 'ma-

[25] YAGO URIs in DBpedia usually consist of a name and some numerical characters.

Table 7
Summary of RDFUnit test cases: Aggregation of errors of the 850 triples.

Test Case Source	TC	Success	Fail	Errors
Automatic	3,376	3,341	65	424
Enriched	1,723	1,660	63	137
Manual	47	7	10	204
Total	5,146	5,008	138	765

jority voting'. Furthermore, in this stage the crowd achieved higher precision (0.5291 for 'majority voting') than in the *Find* stage. Moreover, the 'majority voting' setting obtained 1.0000 for sensitivity, since workers did not produce false negatives, i.e., workers did not classify incorrect triples as correct. Another important result is exhibited by the metric specificity; low values of specificity in this task confirms that the crowd has difficulties when processing triples that are correct generating a large portion of false positives.

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 is 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. This is consistent with the behavior observed in the *Verify* stage of the first workflow.

5.4. Evaluation of (Semi-)Automatic Approaches

We took the same set of resources from DBpedia that were assessed in the crowdsourcing experiments, and executed (semi-)automatic approaches for each studied quality issue. The goal of this study is to gain insights about the type of inconsistencies or errors that can be detected (semi-)automatically, and in which cases human contributions are still beneficial. The obtained results are discussed in the following.

5.4.1. Object Values, Datatypes, and Literals

We used the *Test-Driven Quality Assessment* (TDQA) methodology [26] as our main comparison approach to detect incorrect object values, datatypes and language tags. 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 queries. 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 manually specified as SPARQL queries.

RDFUnit[26] [25] is a tool that implements the TDQA methodology. RDFUnit generates automatic test cases for enabled schemata and checks for common axiom validations. A test is 'successful' when there are no violations of the tested axiom; if violations are found then the test 'fails'. Currently, RDFUnit supports the detection of inconsistencies for *domain* and *range* for RDFS as well as *cardinality*, *disjointness*, *functionality*, *symmetry* and *reflexiveness* for OWL under Closed World Assumption (CWA).

In these experiments, we re-used the same setup for DBpedia used by Kontokostas et al. [26], 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[27]. In addition, we re-used the axioms produced by the ontology enrichment step for DBpedia, as described by Kontokostas et al. [26]. In total, 5,146 tests were run on the 509 (object values) and 341 (datatype/language tags) triples detected as incorrect by workers in the *Verify* stage (Table 5). In particular: 3,376 tests 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 of 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 originated from ontological constraints such as functionality, datatype and domain violations. Common violation instances of ontological constraints were multiple birth/death dates and population values, datatype

[26] http://rdfunit.aksw.org

[27] Schema prefixes as used as defined in Linked Open Vocabularies (http://lov.okfn.org).

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.**
Assymmetric (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**

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), *person height range* (51), *invalid postal codes (warning)* (13), *persons without a birth date (warning)* (51), *persons with death date that should also have a birth date (warning)* (3), *a resource with coordinates should be a dbo:Place (warning)* (16), and *a dbo:Place should have coordinates (warning)* (7). It is worth noting that some of the manual constraints are marked as warnings. Depending on the actual use of the data, these violations could possibly be ignored, taken into consideration or subjected to a moderation or crowdsourcing step for verification.

The *person height range* check resulted in 51 violations. This test case was manually specified as a SPARQL query and is not presented in Table 8. The test case checked wether a person's height is between 0.4 and 2.5 meters. In this specific case, the unit was meters and the values were extracted as centimeter. Thus, although the results appeared semantically valid to a user, they were actually wrong.

A complete direct comparison with our crowdsourcing results was not possible except for 85 wrong datatypes and 13 failed regular expressions (cf. Table 8). However, even in this case it was not possible to provide a precision since RDFUnit runs through the whole set of resources and possibly catches errors for which we did not have a curated gold standard. In an inspection to the outcome of RDFUnit, we observed that RDFUnit was able to identify incorrect triples that were not detected by the LD experts that participated in our contest. The reason for this was that RDFUnit was running beyond the isolated triple level that the LD experts and crowd were evaluating and was checking various combinations of triples. For example, rdfs:domain violations were not reported from the LD experts since for every triple it was required to cross-check the ontology definitions for the evaluated property and the rdf:type statements of the resource. Similar combinations applied for all the other patterns types described in Table 8. Although the experts had the means to access portions of the schema definitions via *TripleCheckMate*, manually validating ontological constraints is a cognitive task which coulds become very difficult since some constrains might require complex combinations of further constraints. Still, the LD experts were able to detect incorrect triples that were not found by RDFUnit. Examples of such inconsistencies are erroneous language tags or incorrect datatypes which are not properly defined in the ontology[28], e.g., dates vs. numbers (dbp:yearOfBirth "1935"^^xsd:integer).

The results of automatically evaluating RDFUnit elucidate the type of inconsistencies or errors that can be identified exploiting the constraints encoded in ontologies. To detect further logical inconsistencies, RDFUnit relies on domain experts to define custom rules, as in our simple example of human height measurements. Still, semantic correctness of triples cannot always be specified as ontology constraints and therefore might require human judgment. In these cases, crowdsourcing mechanisms can be used in combination with tools like RDFUnit to provide more comprehensive solutions for LD quality assessment.

5.4.2. Automatic Baseline to Assess Incorrect Interlinks

We implemented a simple baseline that dereferenced, for each triple, the object of the triple. The baseline then searched for occurrences of the foaf:name of the subject within the dereferenced data. If the number of occurrences was greater or equal than one, i.e., the subject was mentioned at least once, the baseline interpreted the object of the triple as being related to the subject. In this case the link was considered correct. Listing 1 shows the script used to count the number of

[28] And this is very common case for the DBpedia namespace http://dbpedia.org/property/.

times the title of the resource appears in the dereferenced data.

Listing 1: Script for detecting whether the interlink is correct where $a is the triple object and $b is the title of the resource.

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

The script from Listing 1 did not take into consideration the semantics of the links and failed in cases when data dereferenced from objects did not contain backlinks to the issued subject. Another case when the baseline failed was when the objects corresponded to images (via predicates foaf:depiction or foaf:thumbnail), although we configured the baseline to check whether the subject occured in the file name of the image.

In order to compare the baseline with the crowdsourcing approaches (i.e. detection whether the interlinks are correct), we extracted the interlinks from the triples that were involved in both the *Verify* stages of the crowdsourcing experiments with workers.

From both the Verify stages, 223 interlinks each were retrieved. As a result of running this script, we detected a total of 161 and 128 interlinks that were not detected to have the title of the resource in the external web page (link) in the first and second stage respectively. That is, only 48 (in case of the first) and 54 (in case of the second experiment) of the total 223 interlinks each were detected to be correct by this automatic approach. A precision of 0.2296 and 0.2967 was obtained for each of the stages. Thus, the presented baseline illustrated that although some links can be excluded from human judgement, the majority of the examined links could be properly assessed using simple solutions.

6. Final Discussions

Referring back to the research questions formulated in Section 5, our experiments let us identify the strengths and weaknesses of applying crowdsourcing mechanisms for assessing the studied data quality issues, 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. The precision reached by both crowds showed that crowdsourcing is a feasible solution to detect the studied LD quality issues in DBpedia **(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 suggested that the effort 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 were 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 proved to be very skilled at verifying whether literals were written in a certain language. In addition, workers were exceptionally good and efficient at performing comparisons between data values when some contextual information was provided. This was corroborated by the outcome of the "incorrect value" task where workers compared values from DBpedia and Wikipedia.

Furthermore, we were able to detect common cases in which none of the two forms of crowdsourcing we studied seemed to be feasible. The most problematic task for the LD experts was the one about discerning whether a Web page was related to an RDF resource. Although the experimental data did not provide insights into this behavior, we are inclined to believe that this was due to the relatively higher effort required by this specific type of task, which involved checking an additional site outside the *TripleCheckMate* tool. Although the crowd outperformed the experts in *finding* incorrect 'interlinks', the MTurk crowd was 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 suggested 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 automatic 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 $2,294$ assignments 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.

In summary, our experimental results confirm that crowdsourcing-based workflows are a feasible solution for detecting the studied LD quality issues. However, since triples are assessed individually, the scalability of the approach is compromised when issuing large datasets. Therefore, we consider that our proposed approach could reach its full potential when it is combined with automatic approaches in two ways: i) Automatic approaches can help to significantly reduce the number of triples that resort to crowdsourcing; ii) The outcome of the crowd can be used as training sets consumed by automatic approaches to detect quality issues in further portions of a given LD dataset. Building hybrid human-machine architectures will allow for devising efficient and effective solutions for LD quality assessment able to scale up to large datasets.

7. Related Work

We focus on investigating two types of related work: *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 [3]. 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 [28], and competitions such as the yearly Semantic Web Challenge[29] and the European Data Innovator Award[30].

From a process point of view, Villazón-Terrazas and Corcho [55] 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, Luczak-Rösch et al. [33] mapped 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 the work by Siorpaes and Simperl [48]. 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 datasets 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 the work by Simperl et al. [46], 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 *Find-Fix-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 or laymen – to engage with the given data management tasks. This has been subject to several user experience design studies [34,40,45,53,54], which informed the implementation of our crowdsourcing projects, both the contest,

[29] http://challenge.semanticweb.org/

[30] http://2013.data-forum.eu/tags/european-data-innovator-award.html

and the paid microtasks running on Amazon Mechanical Turk. For instance, microtasks have been used for entity linking in ZenCrowd [10] quality assurance, entity resolution in CrowdER [57], ontology alignment in CrowdMap [43] and completing missing values in RDF data in HARE [1].

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 [35,51,56] 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 [47] propose several games that deal with the task of data interlinking, be that in its ontology alignment instance (SpotTheLink [51]), multimedia interlinking (SeaFish [50]) or spotting inconsistencies in data (*WhoKnows?* [56]). Similar ideas are implemented in GuessWhat?! [35], a selection-agreement game which uses URIs from DBpedia, Freebase and OpenCyc as input to the interlinking process. 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 a contest and 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 [12,42].

A similar study is discussed in the work by McCann et al. [36] for ontology alignment. This work 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 [43] 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.

Also in the context of Linked Data management, a human-enabled approach to execute queries against RDF datasets has been proposed. Acosta et al. presented HARE [1], a hybrid SPARQL query processing engine to enhance the quality of query answers. HARE relies on microtask crowdsourcing to complete missing values that are detected during query execution. Experimental results of HARE confirmed that laymen are able to assess RDF data from diverse knowledge domains including Life Sciences. While HARE focuses on data completeness, our approach is tailored for assessing quality issues that affect data accuracy.

7.2. Web Data Quality Assessment

Existing frameworks for quality assessment of the Web of Data, including Linked Data, can be broadly classified as automated [13,18,19,38,39,32], semi-automated [7,15,29,52] and manual [5,37].

In particular, for the quality issues used in our experiments, the work presented by Guéret et al. [19] 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, the framework SWIQA proposed by Fürber and Hepp [18] can be applied for detecting accuracy quality issues including incorrect object values, datatypes and literal. However, in these approaches they either lack specific syntactical rules to detect all of the errors and/or require knowledge of the underlying schema for the user to specify these rules. Other automatic solutions rely on clustering or statistical-based algorithms to detect different quality issues in LD sets [13,32,38,39]. Fleischhacker et al. [13] proposed a two-fold approach that relies on unsupervised outlier detection methods to identify numerical errors in objects of RDF triples. Similarly, Li et al. [32] presented a probabilistic framework that predicts arithmetic relations (equal, greater than, less than) between multiple RDF predicates in order to detect inconsistencies in numerical and date values. Other works have also proposed automatic approaches to improve the quality of LD in terms of completeness and accuracy. In this regard, Paulheim and Bizer presented two algorithms *SDType* [38,39] and *SDValidate* [39] that rely on statistical distributions of predicates and objects in RDF datasets. *SDType* predicts classes of RDF resource thus completing missing values of rdf:type properties. *SDValidate* detects incorrect links between resources within a dataset. These solutions [13,38,39,32] are tailored to detect very specific errors in RDF triples, however, they can be used in combination with our approach to prune RDF triples or quality issues that do not require human assessment.

Semi-automatic approaches to tackled quality assessment have been also proposed. Flemming [15] provides a form-based interface users can specify the SPARQL endpoint and exemplary URIs of the dataset as input and in return she receives an overall quality score. However, in this case, the results are difficult

to interpret and require the user to specify different weights for different quality metrics at each step of the assessment, which makes it cumbersome especially when the user may not know the dataset in much detail. CROCUS [7] is a clustering-based framework that identify outliers at ontologies' instance-level to detect inconsistencies in LD sets. Outliers are then assessed by non-experts denominated quality raters. CROCUS is able to detect violations in cardinality constraints or value ranges. In the context of ontology enriching, Lehmann and Bühmann presented ORE [29], a tool to detect ontology modeling problems. ORE implements reasoning as well as semi-automatic supervised learning to provide suggestions to users (knowledge engineers) for enriching ontologies. Töpper et al. [52] proposed an approach to enrich ontologies with class disjointness as well as property domain and range restrictions. The latter approach is able to detect semantic errors that cannot be detected with syntactic validators or reasoners. The outcome of this approach is a set of suggestions to correct inconsistencies that are processed manually. Unlike CROCUS, the latter two solutions [29,52] might require either domain or ontology experts since implementing changes in ontological constructs could generate further inconsistencies.

In case of manual assessment methodologies or frameworks [5,37], the WIQA quality assessment framework [5] consists of a set of software components for filtering information from the Web using a range of different filtering policies or metrics. In case of Sieve [37], the definition of metrics has to be done by creating an XML file, which contains specific configurations for a quality assessment task. Even though these frameworks introduce useful methodologies to assess the quality of a dataset, the results are difficult to interpret and mandate a considerable amount of user prior knowledge and involvement.

Other studies analyzed the quality of Web [6] and RDF [21] data. The latter study focuses on errors occurred during the publication of LD datasets. Furthermore, a study [22] 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 dataset.

SPARQL Inferencing Notation (SPIN)[31] is a W3C submission aiming at representing rules and constraints on Semantic Web models using SPARQL. The approach described in [17] advocates the use of SPARQL and SPIN for RDF data quality assessment. In a similar way, Fürber et al. [16] 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[32]. *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) [41] and *IBM Resource Shapes*[33].

In summary, our work is situated at the intersection of the previously discussed research areas. In Section 7.1, we explained how crowdsourcing in various forms, e.g., contests, games with a purpose, microtasks, have been successfully applied to resolve diverse aspects of LD management. However, our work studies novel applications of crowdsourcing to detect specific LD quality issues with crowds composed by experts and non-experts. Furthermore, unlike the solutions presented in Section 7.2 for assessing the quality of Web Data, our approach solely relies on human intervention to detect errors in LD.

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 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 with a contest to engage with a community of LD experts. The task of the contest consists in discovering and classifying quality issues of DBpedia resources using the *TripleCheckMate* tool. Contributions obtained through the contest (referring to flawed object values, incorrect datatypes or language tags, and incorrect links) were submitted to Amazon Mechanical Turk (MTurk), where we asked workers to *Verify*

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

[32] http://clarkparsia.com/pellet/icv/

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

them. For the second part of our study, 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 the studied LD issues in DBpedia. In particular, the experiments revealed that (i) lay workers are in fact able to detect certain quality issues with 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. The empirical results of our experiments could serve as a base for further studies in the area of LD quality assessment using human computation. 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.

The methodology proposed in this work is applicable to any LD dataset and can be expanded to cover different types of quality issues. The *TripleCheckMate* tool can be configured to assess other LD datasets using different taxonomies of quality issues. In addition, the proposed algorithms to generate microtasks can also be adapted to build different user interfaces that assist workers in assessing other LD issues. However, the scope of our empirical observations are circumscribed to the studied quality issues within the DBpedia dataset.

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 investigate on how to optimally integrate crowd contributions – by implementing the *Fix* stage – into hybrid human-machine 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 the crowd.

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, E. Simperl, F. Flöck, and M.-E. Vidal. HARE: A hybrid sparql engine to enhance query answers via crowdsourcing. In *To appear in Knowledge Capture K-CAP*, 2015.

[2] 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.

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

[4] 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.

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

[6] 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.

[7] D. Cherix, R. Usbeck, A. Both, and J. Lehmann. CROCUS: cluster-based ontology data cleansing. In *Joint Proceedings of the Second International Workshop on Semantic Web Enterprise Adoption and Best Practice and Second International Workshop on Finance and Economics on the Semantic Web Co-located with 11th European Semantic Web Conference, WaSABi-FEOSW@ESWC 2014, Anissaras, Greece, May 26, 2014.*, 2014.

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

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

[10] 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.

[11] D. E. Difallah, M. Catasta, G. Demartini, P. G. Ipeirotis, and P. Cudré-Mauroux. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 238–247, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[12] 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.

[13] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker, and C. Bizer. Detecting errors in numerical linked data using cross-checked outlier detection. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, editors, *The Semantic Web - ISWC 2014*, volume 8796 of *Lecture Notes in Computer Science*, pages 357–372. Springer International Publishing, 2014.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] J. Juran. *The Quality Control Handbook*. McGraw-Hill, New York, 1974.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] J. Lehmann and L. Bühmann. Ore - a tool for repairing and enriching knowledge bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC2010)*, Lecture Notes in Computer Science, Berlin / Heidelberg, 2010. Springer.

[30] 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.

[31] J. M. Leimeister, M. Huber, U. Bretschneider, and H. Krcmar. Leveraging crowdsourcing: Activation-supporting components for it-based ideas competition. *J. Manage. Inf. Syst.*, 26(1):197–224, July 2009.

[32] H. Li, Y. Li, F. Xu, and X. Zhong. Probabilistic error detecting in numerical linked data. In *Database and Expert Systems Applications*, volume 9261 of *Lecture Notes in Computer Science*, pages 61–75, 2015.

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] H. Paulheim and C. Bizer. Type inference on noisy rdf data. In H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web - ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 510–525. Springer Berlin Heidelberg, 2013.

[39] H. Paulheim and C. Bizer. Improving the quality of linked data using statistical distributions. *Int. J. Semant. Web Inf. Syst.*, 10(2):63–86, Apr. 2014.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] C. Sarasua, E. Simperl, N. F. Noy, A. Bernstein, and J. M. Leimeister. Crowdsourcing and the semantic web: A research manifesto. *Human Computation*, 2015. to appear.

[45] 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.

[46] 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.

[47] K. Siorpaes and M. Hepp. Ontogame: Weaving the semantic web by online games. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 751–766. Springer Berlin Heidelberg, 2008.

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

[49] C. Terwiesch and Y. Xu. Innovation contests, open innovation, and multiagent problem solving. *Manage. Sci.*, 54(9):1529–1543, Sept. 2008.

[50] 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.

[51] 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.

[52] G. Töpper, M. Knuth, and H. Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems I-SEMANTICS*, pages 33–40, 2012.

[53] 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.

[54] 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.

[55] 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.

[56] J. Waitelonis, N. Ludwig, M. Knuth, and H. Sack. Whoknows? - evaluating linked data heuristics with a quiz that cleans up dbpedia. *International Journal of Interactive Technology and Smart Education (ITSE), Emerald*, 8, 2011.

[57] 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.

[58] 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.

[59] 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.