HARE: An Engine for Enhancing Answer Completeness of SPARQL Queries via Crowdsourcing

Maribel Acosta
Karlsruhe Institute of Technology
maribel.acosta@kit.edu

Elena Simperl
University of Southampton
e.simperl@soton.ac.uk

Fabian Flöck
GESIS - Leibniz Institute for the Social Sciences
fabian.floeck@gesis.org

Maria-Esther Vidal
TIB Information Center / Universidad Simon Bolivar
maria.vidal@tib.eu

ABSTRACT

We propose HARE, a SPARQL query engine that encompasses human-machine query processing to augment the completeness of query answers. We empirically assessed the effectiveness of HARE on 50 SPARQL queries over DBpedia. Experimental results clearly show that our solution accurately enhances answer completeness.

CCS CONCEPTS

• **Information systems → Crowdsourcing**; **Resource Description Framework (RDF)**; *Database query processing*;

KEYWORDS

SPARQL; Crowdsourcing; Completeness; Query Execution; RDF

ACM Reference Format:
Maribel Acosta, Elena Simperl, Fabian Flöck, and Maria-Esther Vidal. 2018. HARE: An Engine for Enhancing Answer Completeness of SPARQL Queries via Crowdsourcing. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3184558.3186241

1 INTRODUCTION

As in traditional semi-structured data models, RDF allows for creating datasets that result from integrating multiple, and typically heterogenous and unstructured data sources. In RDF datasets, triples represent positive statements, under the open world assumption. These key aspects of RDF data impose fundamental challenges on deciding data completeness: it is unknown a priori whether a value is actually missing in the dataset, thus negatively impacting completeness of tasks of Linked Data consumption and query processing. To illustrate, consider the motivating example depicted in Figure 1. The SPARQL query (cf. Figure 1b) executed over DBpedia [10] selects movies, including their producers, that have been filmed by Universal Pictures. The query execution returns no producers for 239 out of the 1,461 movies filmed by Universal Pictures. An inspection of the query results reveals that DBpedia has no producers for dbr:Tower_Heist (cf. Figure 1a). However, manually checking external sources, this movie has in fact 3 producers. With cases like this being a common occurrence in RDF datasets, further techniques are needed to improve query answer completeness.

Problem Statement. We tackle the problem of automatically identifying and completing portions of a SPARQL query evaluated over an RDF dataset that yields incomplete results.

Related Work. The Database and Semantic Web communities have extensively studied methods for assuring data quality in traditional databases [13] as well as on web data [2, 7, 17]. Despite all these developments, common sense knowledge acquired from humans may be required for improving effectiveness of automatic methods of data quality assessment [5–7, 16]. In the context of data management, crowdsourcing has been used to design advanced query processing systems that combine human and computational intelligence [9, 12, 15]. The Database community has proposed several human/computer query processing architectures for relational data. In approaches such as CrowdDB [9], Deco [14, 15], Qurk [11], and CrowdOp [8], existing microtask platforms are embedded in query processing systems. These systems provide declarative languages tailored to facilitate an adaptive design of hybrid query execution pipelines. Albeit effective for relational databases, such approaches are less feasible for a Linked Data scenario, which is confronted with autonomous and heterogeneous RDF datasets.

Motivation and Research Questions. Due to the semi-structured nature of RDF and the assumptions of the RDF model (e.g., the open world assumption), the results of crowd-based relational solutions cannot be directly applied to the problem of SPARQL query processing. Therefore, we investigate the following research questions:

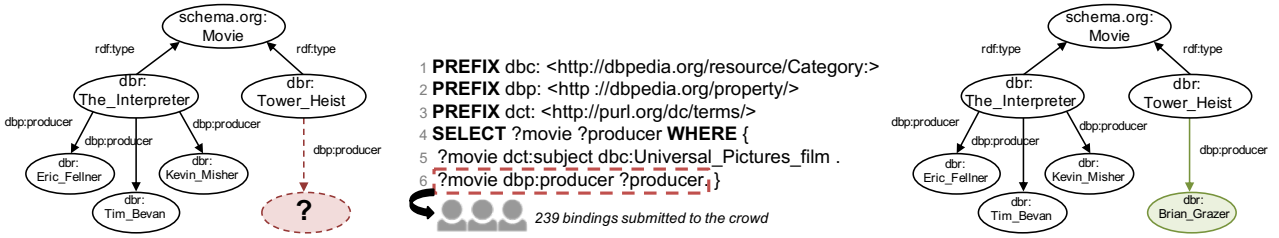
- **RQ1** Can answers of SPARQL queries be completed via hybrid computation without incurring additional complexity in query evaluation?
- **RQ2** Can answer completeness of SPARQL queries be augmented via microtasks?
- **RQ3** What is the impact of exploiting the semantics of RDF resources on crowd effectiveness and efficiency when solving missing values?

Methodology and Contributions. We propose HARE [3, 4], a hybrid SPARQL query engine. HARE integrates crowd knowledge into query processing to enhance query answer completeness. We formally demonstrate the theoretical properties of HARE. In addition, we empirically assess the performance of HARE while evaluating 50 SPARQL queries against DBpedia (version 2014). The novel contributions of this work can be summarized as follows [4]:

- An RDF completeness model that exploits the topology of RDF graphs to estimate the answer completeness of SPARQL

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3186241



(a) The film dbr:The_Interpreter is linked to 3 producers via the dbp:producer predicate, contrary to dbr:Tower_Heist whose producer values are missing in DBpedia.

(b) Query to *select movies (and producers) filmed by Universal Pictures*. This query produces 1,222 results, yet, there are 1,461 movies by Universal Pictures. 239 movies are not associated with producers (e.g., dbr:Tower_Heist).

(c) The crowd stated that *Brian Grazer is a producer of Tower Heist*. Hence, a value for the property dbp:producer of dbr:Tower_Heist is dbr:Brian_Grazer.

Figure 1: Motivating example. (a) Portion of DBpedia for movies and producers. Missing values in the RDF graph are highlighted. (b) SPARQL query executed against DBpedia. Portions of the query (highlighted) affected by missing values are crowdsourced. (c) Crowd answers are mapped into RDF to augment the result of queries.

queries where triple patterns have variables in the subject, predicate, or object position.
- A formal definition of the operations carried out by the proposed microtask manager.
- A fuzzy set semantics of the SPARQL query language; we formally prove the complexity of computing the solution of SPARQL queries under the proposed semantics.
- A query engine able to evaluate SPARQL queries respecting the proposed SPARQL fuzzy set semantics.
- An extensive empirical evaluation that demonstrates the impact of HARE on the effectiveness of the crowd.

2 OUR APPROACH: HARE

We propose HARE, a query engine that automatically identifies portions of a SPARQL query that might yield incomplete results and resolves them via crowdsourcing. The input of the engine is a SPARQL query Q and a quality threshold $\tau \in [0.0; 1.0]$. As part of HARE, we devise an RDF completeness model that estimates missing values in sub-graphs of a dataset, exploiting the topology of the RDF graph and the knowledge collected from the crowd.

The query engine takes into consideration τ, the completeness model, and RDF triples collected from the crowd. Potential missing values are passed to the microtask manager, which contacts the crowd to complete the missing values in the dataset. The HARE engine efficiently combines results retrieved from the dataset with human input to produce the final results for Q.

2.1 RDF Completeness Model

We propose a model to estimate the completeness of portions of RDF datasets. The intuition behind our model is to capture the number of different subjects, predicates, and objects in RDF triples, i.e., the multiplicity of RDF resources. To estimate the completeness of a given resource, the model compares its multiplicity with the multiplicity of other resources that belong to the same classes in the dataset. Then, by comparing the topology of other resources, the model can estimate whether a value is missing for a resource.

For example, consider the RDF graph from Figure 1a. The movie dbr:Tower_Heist contains no values for the predicate dbp:producer. Nonetheless, other resources of the class schema.org:Movie (e.g, dbr:The_Interpreter) have multiplicity 3 for dbp:producer. Based on this information, and considering all the resources of the class, the completeness model can estimate that the movie dbr:Tower_Heist is incomplete with respect to this predicate.

2.2 Representation of the Crowd Knowledge

RDF triples allow for representing positive facts, nonetheless, considering negative knowledge is crucial to model the local closed world assumption. Moreover, using human knowledge effectively demands the representation of negative or even unknown statements: in some cases, human contributors might assert that a statement cannot hold or that they do not know the answer to a question. Therefore, in HARE, the knowledge from the crowd CKB is captured in three knowledge bases:

$$CKB = (CKB^{+}, CKB^{-}, CKB^{\sim})$$

CKB^{+} comprises RDF triples that should belong to the dataset (positive facts). CKB^{-} contains triples that should not exist in the dataset (negative facts). $CKB^{\sim}$ lists all associations that the crowd could not confirm or deny (unknown facts). The crowd knowledge bases are modeled as fuzzy sets, i.e., triples are annotated with a membership degree that represents the confidence of the crowd.

The representation of crowd knowledge as CKB^{+}, CKB^{-}, and $CKB^{\sim}$ allows for modeling contradictions or unknownness in crowd answers. Contradiction about the existence of a value arises when members of the crowd assert that the value exists and does not exist. In HARE, contradictions can be detected by comparing the triples in CKB^{+} and CKB^{-}. Unknownness indicates that the crowd does not know whether a value exists. Statements for which the crowd has declared to be unknowledgeable about are stored in $CKB^{\sim}$.

2.3 Query Engine

The HARE query engine combines information from RDF datasets, the crowd knowledge bases containing curated results of prior crowdsourced tasks, and the crowd itself. Because RDF data collected from the crowd knowledge bases and the crowd is not necessarily precise, our approach uses fuzzy RDF to capture multiple degrees of vagueness and imprecision when combining data from crowd-based sources. In [4], we propose a fuzzy set semantics of SPARQL queries such that correct data from RDF datasets and vague data from crowd knowledge bases can be merged during

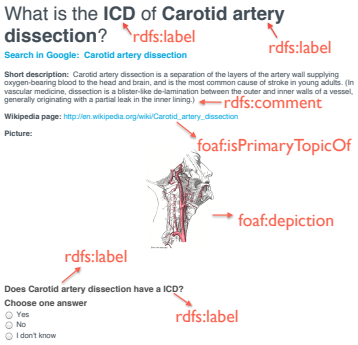


Figure 2: Microtask generated for crowdsourcing the triple pattern (dbr:Carotid_artery_dissection, dbp:icd, ?icd).

SPARQL query processing. Furthermore, in [4] we demonstrated the theorems that derived the following corollary.

Corollary 2.1. *The complexity of computing the mapping set of a SPARQL query under fuzzy set semantics is the same as when it is computed under set semantics.*

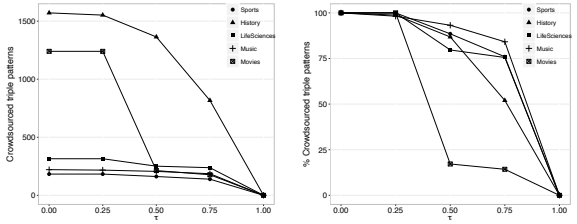
For the HARE query engine, we propose an efficient algorithm that executes BGPs of SPARQL queries under fuzzy set semantics. During query execution, the algorithm combines data from an RDF dataset D and a crowd knowledge base CKB that contains fuzzy sets of RDF data. In HARE, all triples in D are assumed to have membership degree equal to 1.0, since they are assumed to be correct. Furthermore, the query engine considers the completeness model and knowledge captured from the crowd. When the evaluation of a triple pattern leads to incomplete answers, the query engine verifies if the crowd can provide the missing mappings. HARE aims at crowdsourcing triple patterns where the crowd exhibits: i) high confidence values in positive or negative facts, or ii) high levels of contradiction but low unknownness.

2.4 Microtask Manager

This component creates human tasks from triple patterns and collects the crowd answers. The microtask manager is composed of the user interface generator and the microtask executor.

The user interface generator receives as input the triple patterns to be crowdsourced. This component is able to generate interfaces for triple patterns with at most one variable. In addition, this component exploits the semantics of RDF resources in triple patterns to build rich human-readable interfaces to RDF data. The human-readable information is obtained by dereferencing URIs in the triple pattern. The user interface generator displays the values (if available) of different properties of RDF resources (cf. Figure 2). In this way, HARE exploits the semantic descriptions of resources and includes further properties in the microtasks. The more properties to describe the resources are included in the microtasks, the less ambiguous the task is. Providing details like these in microtasks has also proven to assist the crowd in providing right answers [5].

The microtask executor submits the human tasks created by the user interface generator to the crowdsourcing platform. Answers provided by the crowd in each task are retrieved by the microtask executor and processed in order to update the crowd knowledge bases CKB accordingly.



(a) Number of crowdsourced triple patterns per domain

(b) % of crowdsourced triple patterns w.r.t. intermediate results

Figure 3: Effectiveness of the RDF completeness model.

3 EXPERIMENTAL STUDY

Query Benchmark. We designed a benchmark of 50 queries[1] by analyzing triple patterns answerable by the DBpedia dataset (version 2014). The benchmark includes five categories with 10 queries each to study the crowd behavior across different domains: Sports, Music, Life Sciences, Movies, and History.

Gold Standard. We built a gold standard D^* of missing answers by removing portions of the dataset. Depending on the query, the gold standard contains between 8% and 97% of the query answer.

Implementation. HARE is implemented in Python 2.7.6. and we use CrowdFlower [1] as the crowdsourcing platform. We implemented two variants of our approach which generate different microtasks: **HARE** that exploits the semantics of resources as described in Section 2.4, and **HARE-BL** is a baseline approach that simply substitutes URIs with labels in the microtasks.

Crowdsourcing Configurations. We asked workers to solve a maximum of 4 RDF triples per task. The monetary reward was 0.07 US dollars per task. We collected at least 3 answers per task.

In this study, we submitted 1,004 triple patterns to the crowd with HARE and HARE-BL. In total, we collected 3,163 crowd answers.

3.1 HARE Crowdsourcing Capabilities

We measure the number of triple patterns that are crowdsourced when executing the benchmark queries with HARE for different values of the threshold τ. Figure 3a shows that the number of crowdsourced triple patterns differs per knowledge domain. In certain domains (such as History and Movies) the benchmark queries produce a large amount of results with respect to queries from other domains. Figure 3a further indicates that in domains where queries produce large amount of results, HARE – based on the estimations of the completeness model – also crowdsources a large number of triple patterns. These results illustrate how the value of τ impacts the number of crowdsourced triple patterns: the higher the value of τ the lower is the requested completeness of the answer.

We then measure the percentage of crowdsourced triple patterns with respect to the size of intermediate results (Figure 3b). For $\tau \geq 0.50$, the completeness model is able to prune the number of tasks submitted to the crowd. In particular, in domains where the benchmark queries produce a large number of intermediate results – such as History and Movies –, the completeness model reduces the number of crowdsourced triple patterns considerably (for $\tau = 0.75$).

[1] https://sites.google.com/site/hareengine

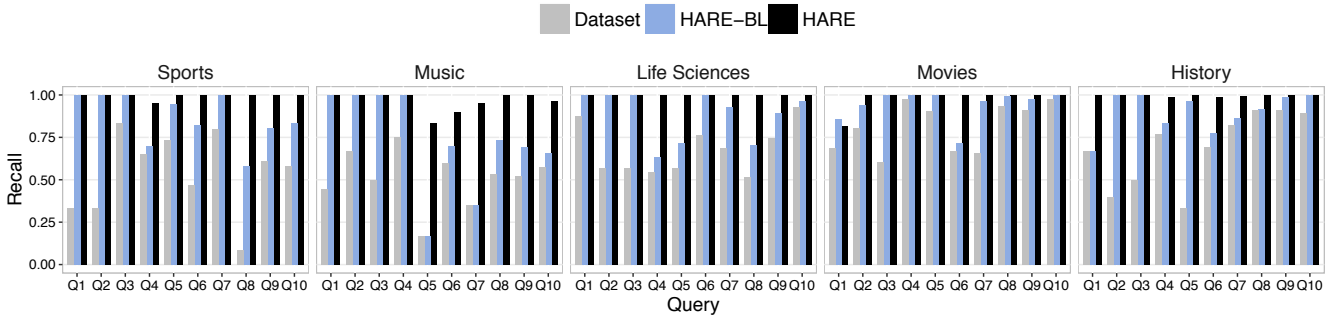
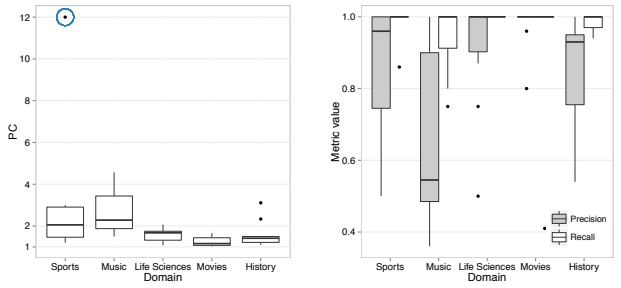


Figure 4: Recall: Query answer completeness (y-axis) obtained with DBpedia (Dataset) and our approaches HARE and HARE-BL per query (x-axis). HARE consistently outperforms the other approaches in all benchmark queries.

3.2 Completeness of Query Answers

To measure the effectiveness of our solution, we compute the proportion of completeness (PC) per query. PC corresponds to the ratio of answers produced by HARE to the answers when the same query is executed only against the dataset. Figure 5a depicts the PC values achieved by HARE per knowledge domain. In all domains, the minimum PC values are higher than 1.0 indicating that HARE always increased the number of answers in all SPARQL queries. It is important to highlight that PC values are affected by the completeness of the dataset. This is the case, for instance, in DBpedia in the domains of Life Sciences and Movies, which exhibit high completeness. Therefore, the PC values achieved in these domains are not as high as for other knowledge domains in DBpedia.

In addition, we report on the recall values obtained with HARE, HARE-BL, and when the queries are executed directly over the dataset (cf. Figure 4). We can observe that the recall when querying DBpedia varies among queries and knowledge domains. This indicates that completeness in DBpedia is heterogenous among different sub-graphs, in this case, represented by different knowledge domains. These results support the importance of taking into consideration the local completeness of resources. Furthermore, HARE and HARE-BL are able to improve on recall, which suggests that our RDF completeness model is able to capture the skewed distribution of values in real-world datasets. Lastly, HARE outperforms the other approaches suggesting that semantically enriched interfaces enable the crowd to provide correct answers.



(a) Proportion of completeness (PC) achieved by HARE.

(b) Precision and recall of crowd answers per domain.

Figure 5: HARE effectiveness.

3.3 Quality of Crowd Answers

In this study, we compute precision and recall of the triples retrieved from the crowd with respect to the gold standard D^*.

Figure 5b reports on the aggregated results of precision and recall of crowd answers obtained with HARE. It can be observed that precision values fluctuate over the knowledge domains. The lowest performance in terms of precision is obtained in the Music domain, where the median is 0.55. Still, the high value of the third quartile in the Music domain indicates that most of the precision values range from 0.55 to 0.90. Overall, the median precision values of HARE in the other domains are greater than 0.93. In turn, recall values are consistently high with median equal to 1.0.

HARE microtasks assisted the crowd in reaching perfect precision and recall scores in 30 out of 50 SPARQL queries. These experiments confirm that exploiting the semantics of RDF resources allows the crowd to effectively solve missing RDF values which, in turn, enhances the answer completeness of SPARQL queries.

4 CONCLUSIONS AND OUTLOOK

We presented HARE, a SPARQL query engine that relies on an RDF completeness model and knowledge collected from the crowd to complete missing values. We conducted an experimental study using 50 queries over DBpedia. Based on the empirical results, we can answer the research questions formulated in Section 1.

Answer to RQ1. We formally demonstrate that HARE solves the problem of identifying and evaluating SPARQL sub-queries that yield missing values in polynomial time.

Answer to RQ2. The crowd reached via microtasks resolves missing values in RDF graphs with high precision and recall which, in turn, increases the completeness of SPARQL queries. Nonetheless, there are predicates for which the crowd does not perform well.

Answer to RQ3. We compare the crowd behavior when using microtasks built with and without semantics. We observe a significant difference in crowd performance in microtasks with and without semantics. Our results confirm that semantically enriched microtasks increase the quality and efficiency of crowd answers.

Future work could study other models to accurately capture crowd answer reliability. Another important research direction is to study the impact of – instead of triple-based – more complex microtasks against the crowd in terms of accuracy and cost.

REFERENCES

[1] 2017. CrowdFlower. (2017). http://crowdflower.com

[2] Ziawasch Abedjan, Toni Grütze, Anja Jentzsch, and Felix Naumann. 2014. Profiling and mining RDF data with ProLOD++. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*. 1198–1201.

[3] Maribel Acosta, Elena Simperl, Fabian Flöck, and Maria-Esther Vidal. 2015. HARE: A Hybrid SPARQL Engine to Enhance Query Answers via Crowdsourcing. In *Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015, Palisades, NY, USA, October 7-10, 2015*. 11:1–11:8.

[4] Maribel Acosta, Elena Simperl, Fabian FlÃűck, and Maria-Esther Vidal. 2017. Enhancing answer completeness of SPARQL queries via crowdsourcing. *Web Semantics: Science, Services and Agents on the World Wide Web* (2017). https://doi.org/10.1016/j.websem.2017.07.001

[5] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. 2013. Crowdsourcing Linked Data Quality Assessment. In *ISWC*. 260–276.

[6] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing. In *SIGMOD*. 1247–1261. https://doi.org/10.1145/2723372.2749431

[7] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 601–610.

[8] Ju Fan, Meihui Zhang, Stanley Kok, Meiyu Lu, and Beng Chin Ooi. 2015. CrowdOp: Query Optimization for Declarative Crowdsourcing Systems. *IEEE Trans. Knowl. Data Eng.* 27, 8 (2015), 2078–2092.

[9] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. 2011. CrowdDB: answering queries with crowdsourcing. In *SIGMOD*. 61–72.

[10] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. DBpedia–A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* (2014).

[11] Adam Marcus, David R. Karger, Samuel Madden, Rob Miller, and Sewoong Oh. 2012. Counting with the Crowd. *PVLDB* 6, 2 (2012), 109–120.

[12] Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. 2011. Crowdsourced Databases: Query Processing with People. In *CIDR*. 211–214.

[13] Felix Naumann. 2002. *Quality-Driven Query Answering for Integrated Information Systems*. Lecture Notes in Computer Science, Vol. 2261. Springer. https://doi.org/10.1007/3-540-45921-9

[14] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom. 2012. Deco: A System for Declarative Crowdsourcing. *PVLDB* 5, 12 (2012), 1990–1993.

[15] Hyunjung Park and Jennifer Widom. 2013. Query Optimization over Crowdsourced Data. *PVLDB* 6, 10 (2013), 781–792.

[16] Hyunjung Park and Jennifer Widom. 2014. CrowdFill: collecting structured data from the crowd. In *SIGMOD*. 577–588.

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