

Managing and Consuming Completeness Information for Wikidata Using COOL-WD

Radityo Eko Prasajo^(✉), Fariz Darari, Simon Razniewski, and Werner Nutt

KRDB, Free University of Bozen-Bolzano, 39100, Italy
rprasajo@unibz.it

Abstract. Wikidata is a fast-growing, crowdsourced, and entity-centric KB that currently stores over 100 million facts about more than 21 million entities. Such a vast amount of data gives rise to the question: How complete is information in Wikidata? It turns out that there is no easy answer since Wikidata currently lacks a means to describe the completeness of its stored information, as in, Which entities are complete for which properties?

In this paper, we discuss how to manage and consume meta-information about completeness for Wikidata. Due to the crowdsourced and entity-centric nature of Wikidata, we argue that such meta-information should be simple, yet still provide potential benefits in data consumption. We demonstrate the applicability of our approach via COOL-WD (<http://cool-wd.inf.unibz.it/>), a completeness tool for Wikidata, which at the moment collects around 10,000 real completeness statements.

Keywords: data completeness, meta-information, completeness analytics, query completeness, Wikidata, entity-centric KBs

1 Introduction

Given its open and distributed nature, data semantics on the Semantic Web has always been problematic. Generally the open-world assumption (OWA) is taken [7], i.e., datasets are assumed only to describe a subset of reality. However, the OWA leaves users clueless as to whether present data describes *all* data about a certain topic, and is particularly ill-suited for non-monotonic query languages. SPARQL, for instance, simply computes query answers based on the closed-world assumption (CWA), even though these may be incorrect under the OWA.

In fact, the data modeling community observed early on the need for a more fine-grained setting between the OWA and CWA, called the partial closed-world assumption (PCWA) [11,8,14]. The PCWA uses specifications called table-completeness (TC) statements for distinguishing parts of a database that should be considered under CWA from those that should be considered under OWA. This knowledge can then be propagated to query answers, thus allowing to assess whether a query answer is complete, or potentially not. The PCWA has also been proposed for the Semantic Web [3].

Yet, the theoretical foundations so far have not reached practice. Up to now, users can only write completeness information manually in RDF, and would need to publish and link them on their own, in order to make them available. Similarly, users interested in making use of completeness statements have no central reference for retrieving such information so far. We believe that the lack of tools supporting both ends of the data pipeline, production and consumption, is a major reason for the PCWA not being adapted on the Semantic Web so far.

From the data quality perspective, data completeness is an important quality aspect.¹ When data is verified to be complete, decisions taken from the data will be more justifiable. As a motivating scenario, let us consider the data about Switzerland (<https://www.wikidata.org/wiki/Q39>) on Wikidata.

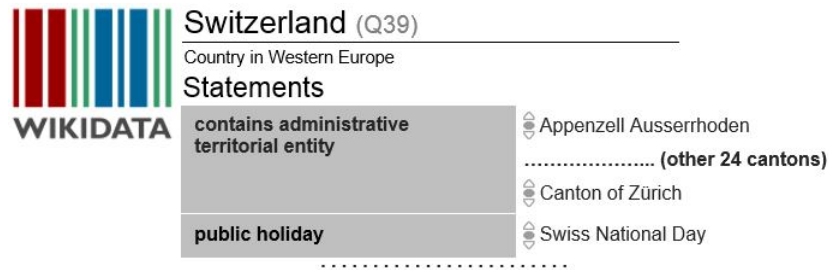


Fig. 1. Wikidata page of Switzerland

At the moment of the writing, Wikidata contains 26 cantons of Switzerland from Appenzell Ausserrhoden to Canton of Zürich. According to the official page of the Swiss government,² there are exactly these 26 Swiss cantons as stored in Wikidata. Therefore, as opposed to the public holidays, whose completeness is still unclear, the data about Swiss cantons in Wikidata is actually complete. However, Wikidata currently lacks support for expressing completeness information, thus limiting the potential consumption of its data (e.g., assessing query completeness or completeness analytics). In general, not only Wikidata, but also other entity-centric KBs (e.g., DBpedia, YAGO) have such a limitation. In this work, we focus on Wikidata as an example use case of the idea of adding completeness information and enhancing data consumption with such completeness information as it is open, fast-growing, and one of the most actively-edited KBs.

Contributions. In this paper we present COOL-WD, a completeness tool for Wikidata. With COOL-WD, end users are provided with web interfaces (available both via COOL-WD external tool or COOL-WD integrated Wikidata gadget) to view completeness information about Wikidata facts, and to perform the creation of completeness information without writing any RDF syntax. Similarly,

¹ <http://www.wired.com/insights/2013/05/the-missing-vs-in-big-data-viability-and-value/>

² <https://www.admin.ch/opc/de/classified-compilation/13.html>

a query interface is provided that allows one to assess whether query answers can be considered under the OWA or CWA, and Linked Data API to access completeness statements as RDF resources. Finally, COOL-WD provides a centralized platform for collecting completeness information. It comes prepopulated with over 10,000 completeness statements, which are available for download.³

Related Work. Knowledge base quality has received attention since long, facing general issues [9,22], entity linking issues [13], or inconsistency issues [19,18]. Completeness itself is a quality aspect that relates to the breadth, depth, and scope of information contained in the data [21]. A number of approaches aimed to achieve a higher degree of data completeness [12,2,1]. Yet in these approaches, no assumption was made about completeness wrt. the real-world information.

The idea for a declarative specification of data completeness can be traced back to 1989 [11]. Various works have built upon this, most lately leading to a tool called MAGIK [17], which allows one to collect expressive completeness information about relational databases, and to use it in query answering. In [3], the framework underlying MAGIK [14] has been ported to the Semantic Web, from which a prototype to gather schema-level completeness information over generic RDF data sources, called CORNER, was then developed [4]. What are still missing from this prototype are practical considerations: completeness statements were too complex for general users and there was no tight integration with the data to which completeness statements are given due to their generality. In [5], a practical fragment of completeness statements, called SP-statements, was first identified. The work focused more on the formal aspects of SP-statements, especially on how to characterize fine-grained query completeness assessment from such statements. Now, in this paper practical aspects of SP-statements are discussed, including how to make them available as Linked Data resources.

2 Background

2.1 Wikidata

We focus on the Wikidata [20] knowledge base, which provides its data in RDF. Wikidata is a fast-growing,⁴ entity-centric, volunteered KB with currently 16,000 active users⁵ that aims to complement the Wikipedia effort with structured data. In Wikidata, entities and properties are identified by internal IDs with the format Q(number) for entities (e.g., Q39 for Switzerland), and P(number) for properties (e.g., P150 for “contains administrative territorial entities”). As an illustration, Wikidata stores the triple (*Q39, P150, Q12724*),⁶ meaning that Switzerland contains Ticino (Q12724) as one of its administrative territorial entities.

³ <http://completeness.inf.unibz.it/rdf-export/>

⁴ <https://tools.wmflabs.org/wikidata-todo/stats.php>

⁵ <https://www.wikidata.org/wiki/Wikidata:Statistics>

⁶ We omit namespaces for readability. Later on, we also refer to resources via their labels.

2.2 SP-Statements

Now we describe SP-statements and show how they can enrich entity-centric KBs (including Wikidata) with completeness information. Furthermore, we discuss how to provide them as Linked Data resources.

Motivation. Popular knowledge bases (KBs) like Wikidata, DBpedia, and YAGO, provide their information in an entity-centric way, that is, information is grouped into entities in such a way that each entity has its own (data) page, showing the entity's property-value pairs. Those KBs in general follow the open-world assumption: it is unknown whether an entity is complete for the values of a property. Yet, in practice, oftentimes entities are complete for specific properties (e.g., all children of Obama, all crew members of Apollo 11, etc). Having *explicit* completeness information increases the informativeness of an entity: not only do we know that some property values are valid for an entity, but also we know that those are *all* the values.

SP-statements are statements about the completeness of the set of values of a property of an entity. Such statements capture exactly the idea of providing completeness information whose structure is as close as possible to the SPO-structure underlying entity-centric KBs. Moreover, given their simplicity, SP-statements can be provided in the crowdsourcing way, which again would fit into how entity-centric KBs are typically constructed (e.g., Wikidata). Furthermore, since SP-statements are basically (quality) metadata about KBs, they may also complement existing approaches to describe KBs like VoID⁷ and DCAT.⁸ Later in Section 5, we will also show how SP-statements pave the new way toward consuming Linked Data.

Syntax and Semantics. Formally, an SP-statement is written as a pair (s, p) where both s and p are URIs. A graph (or dataset) G having an SP-statement (s, p) would mean that all the values of the property p of the entity s in the real-world are captured in G , that is, for a hypothetical ideal graph $G' \supseteq G$ where all kinds of information are complete, G' does not contain new information about the property p of s . For example, an SP-statement for all cantons⁹ in Switzerland can be written as follows: $(\textit{Switzerland}, \textit{canton})$. Stating that Wikidata has this SP-statement implies that all the cantons of Switzerland in reality are recorded in Wikidata. Observe that SP-statements can also be used to state the non-existence of a property of an entity. To do this is simply by assigning an SP-statement (s, p) over a graph in which there is no corresponding triple (s, p, o) for any URI or literal o . For example, to say that Wikidata has the SP-statement $(\textit{elizabethI}, \textit{child})$ would be the same as to say that Elizabeth I had no children since Wikidata contains no children of Elizabeth I.

⁷ <https://www.w3.org/TR/void/>

⁸ <https://www.w3.org/TR/vocab-dcat/>

⁹ A canton is a direct administrative subdivision of Switzerland.

SP-Statements as Linked Data Resources. Having motivated and formalized SP-statements, now we want to make them available in practice, especially by following the Linked Data principles.¹⁰ Therefore, SP-statements should be identifiable by URIs and accessible in RDF. To improve usability, URIs for SP-statements should indicate which entity is complete for what property. For example, in our COOL-WD system later on, we identify the SP-statement (*Switzerland, canton*) with the URI <http://cool-wd.inf.unibz.it/resource/statement-Q39-P150>, indicating the entity Switzerland (Wikidata ID: Q39) and the property “contains administrative territorial entities” (Wikidata ID: P150).

Next, looking up an SP-statement’s URI must give a description of the statement. Thus, we provide an RDF modeling of SP-statements. We divide the modeling into two aspects: core and provenance. The core aspect concerns the intrinsic aspect of the statement, whereas the provenance aspect deals with the extrinsic aspect of the statement, providing information about the generation of the statement. The core aspect consists of the type of the resource, the subject and predicate of the SP-statement, and the dataset to which the statement is given (in the scope of this paper, the dataset is Wikidata). The provenance aspect consists of the author of the statement, the timestamp when the statement is generated, and the primary reference of the statement. For the core aspect, we developed our own vocabulary, available at <http://completeness.inf.unibz.it/sp-vocab>. For the provenance aspect, to maximize interoperability we reused the W3C PROV ontology.¹¹ The following is an RDF modeling of the SP-statement “Complete for all cantons in Switzerland” for Wikidata.

```

@prefix wd: <http://www.wikidata.org/entity/> .
@prefix spv: <http://completeness.inf.unibz.it/sp-vocab#> .
@prefix coolwd: <http://cool-wd.inf.unibz.it/resource/> .
@prefix wdt: <http://www.wikidata.org/prop/direct/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

wd:Q2013 spv:hasSPStatement coolwd:statement-Q39-P150 . # Q2013 = Wikidata
coolwd:statement-Q39-P150 a spv:SPStatement ;
  spv:subject wd:Q39 ; # Q39 = Switzerland
  spv:predicate wdt:P150 ; # P150 = canton
  prov:wasAttributedTo [ foaf:name "Fariz Darari" ;
    foaf:mbox <mailto:fariz.darari@stud-inf.unibz.it> ] ;
  prov:generatedAtTime "2016-05-19T10:45:52"^^xsd:dateTime ;
  prov:hadPrimarySource
<https://www.admin.ch/opc/en/classified-compilation/19995395/index.html#a1> .

```

In the RDF snippet above, we see all the core and provenance aspects of the SP-statement for Wikidata of all cantons in Switzerland with self-explanatory property names. Having such snippets would also provide the possibility to export SP-statements about a dataset into an RDF dump, which may then be useful for data quality auditing or completeness analytics purposes.

¹⁰ <https://www.w3.org/DesignIssues/LinkedData.html>

¹¹ <http://www.w3.org/ns/prov>

3 Creating Completeness Information

In general, one can imagine that SP-statements could either originate from (i) KB contributors, (ii) paid crowd workers, or (iii) web extraction [10]. While our focus with COOL-WD is to provide a tool for the first purpose, we used also other methods to pre-populate COOL-WD with completeness information.

KB Contributors. Wikidata already provides a limited form of completeness statements: no-value statements, that is, assertions stating that for a certain subject-predicate pair, no values exist (e.g., Elizabeth I had no children). We imported about 7600 no-value statements from Wikidata¹² as completeness statements. They directly translate to completeness information. The top-three properties used in no-value statements are “member of political party” (12%), “taxon rank” (11%), and “follows” (11%). The properties “spouse”, “country of citizenship”, and “child” are among the top-15.

Paid Crowd Workers. We imported around 900 SP-statements created for work on completeness rule mining [6], which were created using crowdsourcing. Regarding the crowdsourcing done in that work, it is noteworthy that it comes with issues. The first is the price, about 10 cents per statement. The other is, that crowd workers did not truly provide completeness assertions, instead, they were asked, whether they could find additional facts on a limited set of web-pages. Truly asking crowd workers for checking for evidence for completeness was deemed too difficult in that work.

Web Extraction. We imported about 2200 completeness assertions for the *child* relation that were created in [10] via web extraction. These statements are all about the “child” property in Wikidata, and were generated as follows: The authors manually created 30 regular expressions that were used to extract information about the number of children from biographical articles in Wikipedia. For instance, the pattern “X has Y children” would match the phrase “Obama has two children and lives in Washington”, and can thus be used to construct the assertion that Obama should have exactly two children. In total, the authors found about 124,000 matching phrases, of which, after filtering some low-quality information, about 84,000 phrases that had a precision of 94% were retained. For each of these 84,000 assertions, it was then checked whether the asserted cardinality matched the one found in Wikidata. If that is the case, it was then concluded that Wikidata is complete wrt. the children of the person. For instance, for Obama one truly finds two children in Wikidata, and thus, assuming the correctness of the phrase in Wikipedia, can conclude that Wikidata is complete.

¹² https://www.wikidata.org/wiki/Help:Statements#Unknown_or_no_values

4 COOL-WD: A Completeness Tool for Wikidata

We developed COOL-WD, a web-based completeness tool for Wikidata, that provides a way to annotate complete parts of Wikidata in the form of SP-statements. COOL-WD focuses on the direct-statement fragment of Wikidata, in which neither references nor qualifiers are being used. A COOL-WD user is able to view any Wikidata entity that is annotated with SP-statements for all of its properties. A complete property of an entity is denoted by a green checkmark, while a possibly incomplete one is denoted by a gray question mark. A user can add a new SP-statement for a property of an entity by clicking on the gray question mark. In Section 5, we discuss several ways to consume completeness statements in COOL-WD.

In providing its features, COOL-WD maintains real time communication with Wikidata. On the client side, user action like entity search is serviced by MediaWiki API¹³ calls towards Wikidata, while on the server side the COOL-WD engine retrieves entity and property information via SPARQL queries over the Wikidata SPARQL endpoint¹⁴. User-provided SP-Statements are stored in a specialized database. The engine retrieves SP-statements from the database to annotate the entities and properties obtained from the Wikidata SPARQL endpoint with completeness information, and then sends them to the user via a HTTP connection. The engine also manipulates the DB whenever a user adds a new SP-statement.

Hardware and system specification. Our web server and database server run on separate machines. The web server is loaded with 16GB of virtual memory and 1 vCPU of 2.67GHz Intel Xeon X5650. It runs on Ubuntu 14 and uses Tomcat 7 and Java 7 for the web services. The database server has 8GB of virtual memory and 1 vCPU of 2.67GHz Intel Xeon CPU X5650, running on Ubuntu 12 and using PostgreSQL 9.1.

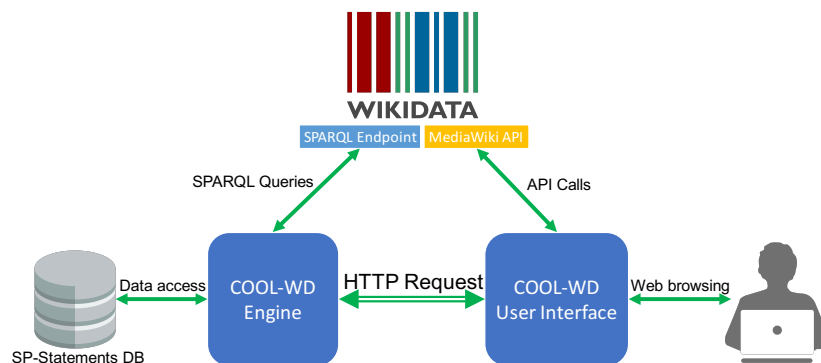


Fig. 2. System architecture of COOL-WD

¹³ <https://www.wikidata.org/w/api.php>

¹⁴ <https://query.wikidata.org/bigdata/namespace/wdq/sparql>

COOL-WD Gadget. Another way is to directly add completeness statements from inside Wikidata, using a Wikidata user script that we created.¹⁵ To activate the script, a user needs a Wikimedia account. Then, she has to import it to her common.js page at [https://www.wikidata.org/wiki/User:\[wikidata_username\]/common.js](https://www.wikidata.org/wiki/User:[wikidata_username]/common.js). Basically, the script makes API requests over our own completeness server that provides a storage service for completeness statements.

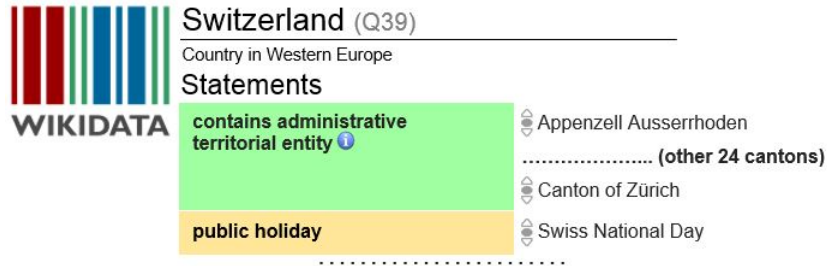


Fig. 3. COOL-WD Gadget: The green box indicates completeness, while the yellow box indicates potential incompleteness

Figure 3 shows that the property box of “contains administrative territorial entity” is colored green, which indicates completeness. The information icon “(i)”, when clicked, provides the reference URL, the Wikidata username, and the timestamp of the completeness statement. Note that for properties not yet known to be complete, they will be colored yellow. To add a completeness statement one simply clicks the yellow property box. To add a reference URL is simply by clicking the information icon and add the URL in the provided form.

5 Consuming SP-Statements

The availability of completeness information of the form SP-statements opens up novel ways to consume data on Wikidata, realized in COOL-WD.

5.1 Data Completion Tracking

The most straightforward impact of completeness statements is that creators and consumers of data become aware of their completeness. Thus, creators know where to focus their efforts, while consumers become aware of whether consumed data is complete, or whether they should take into account that data may be missing, and possibly should do their own verification, or contact other sources.

5.2 Completeness Analytics

Having completeness statements allows us to analyze how complete an entity is compared to other similar entities. For example, for some cantons in Switzerland,

¹⁵ <https://www.wikidata.org/wiki/User:Fadirra/coolwd.js>

Wikidata has complete information about their official languages, but it may not for the other cantons. A Wikidata contributor could exploit this kind of information to spot some entities that are less complete than other similar ones, then focus their effort on completing them.

In COOL-WD, a class of similar entities is identified by a SPARQL query. For example, the class of all cantons of Switzerland consists of the entities returned by the query `SELECT * WHERE { wd:Q39 wdt:P150 ?c }`, where Q39 is the Wikidata entity of Switzerland and P150 is the Wikidata property “contains administrative territorial entity.” A user may add a new class by specifying a valid SPARQL query for the class. Then, COOL-WD would populate all possible properties of the class as a union of properties of each entity of the class. The user would then asked to pick some property that they feel to be important for the class. Now, suppose we pick “official language” and “head of government” as important properties for the cantons of Switzerland, and suppose that we have only the following SP-statements: *(Bern, lang)*, *(Geneva, lang)*, *(Ticino, lang)*, *(Zurich, lang)*, *(Bern, headOfGov)*. Figure 4 shows how COOL-WD displays such analytic information. A user then can see that Wikidata is verified to have complete information about official languages only for 4 out of 26 cantons of Switzerland (15.38 %), which means that the remaining 22 cantons are possibly less complete than the four. Among the four, Wikidata has also complete information for the head of government of Canton of Bern, only one out of the 26 cantons. Using this information, a contributor is able to focus on checking the completeness of the language of the remaining 22 cantons and the head of government of the 25 cantons, whether Wikidata has already complete information about them, so some completeness statements should be added, or Wikidata has incomplete information, so they should add the missing information.


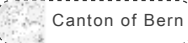

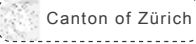

Class name	#Objects	Property	Completeness percentage	Complete entities
Cantons of Switzerland	26	official language	15.38%	 Canton of Geneva  Canton of Bern  Ticino  Canton of Zürich Show less
Cantons of Switzerland	26	head of government	3.85%	 Canton of Bern

Fig. 4. An overview of the completeness analytics feature. Clicking on the class name shows a more detailed analytics of the class.

5.3 Query Completeness Assessment

With explicit completeness information over data comes the possibility to assess query completeness. Intuitively, queries can be answered completely whenever they touch *only* the complete parts of data. We focus on tree-shaped queries, which include also path and star queries that are often used in practice [16], where the root and edges are URIs. Such tree-shaped queries are suitable for

exploring facts about an entity, for instance, who are Obama’s children and spouse, and for his children, what are their schools. In COOL-WD, we implemented an algorithm for completeness assessment as described in [5] and further extended it with a diagnostics feature: depending on whether a query can be guaranteed to be complete or not, users may also see either all SP-statements, including their provenance (i.e., author, timestamp, and reference URL), contributing to the query completeness, or a missing SP-statement as a cause of no completeness-guarantee.

Let us give an illustration on how query completeness assessment works. Consider the query “give all languages of all cantons in Switzerland.” With no completeness information, there is no way to know whether the query answer is complete: it might be the case that a canton, or a language of a canton, is missing.

Now, suppose we have the following SP-statements: $(Switzerland, canton)$, $(Aargau, lang)$, \dots , $(Zurich, lang)$. The statements ensure the completeness of all cantons in Switzerland, and for each canton of Switzerland from Aargau to Zurich, the completeness of all languages. With such completeness information, our query can be answered completely in the following way. The query basically asks for two parts: all cantons of Switzerland, and then for each canton, all of their languages. For the first part, its completeness can be guaranteed by the first statement above. Therefore, by instantiating that complete part, we now need to be complete for the following set of queries: all the languages of Aargau, \dots , all the languages of Zurich. For each of the language queries, we know from the corresponding language SP-statements above that they can be answered completely. Therefore, our whole query can then be answered completely. Additionally, users would see all those SP-statements (and their provenance) that contribute to the query completeness. In contrast, suppose that we do not have the SP-statement $(Zurich, lang)$. In this case, the query cannot be guaranteed to be complete, and COOL-WD reports that $(Zurich, lang)$ is missing.

Experimental Evaluation. To show the feasibility of query completeness assessment over COOL-WD, we conducted an experimental evaluation based on real entities in Wikidata. The experimental setup is as follows. First, we take four various Wikidata classes: country (Q6256), software (Q7397), animated film (Q202866), and political party in Spain (Q6065085). Next, we obtain randomly 100 entities for each class. Then, we generate tree-shaped queries by assigning each entity as the root and traversing the entity over its randomly-chosen properties. We distinguish query generation cases based on the maximum width and depth of the traversal: w2d1, w3d1, w4d1, w5d1, w1d2, w1d3, w1d4, w1d5, w2d2, w3d2, and w2d3, where “w” is the width and “d” is the depth. As each case generates 400 queries, there are in total 4,400 queries. Next, we generate SP-statements from the queries, again by traversing and instantiating each triple pattern of the queries from the root in a top-down manner, taking the subject and predicate of the instantiated triple pattern along the way. Such a generation, however, makes all queries complete. To have a variety of success and failure cases of completeness, we thus randomly remove 50% of the statements.

In total, there are around 11,000 SP-statements generated. In the experiment, we measure the query completeness assessment time and query evaluation time, and group the results by query generation case.

From the experiment results, we can draw several observations. On an absolute scale, the average completeness assessment time of all cases ranges from 74 ms to 309 ms. Also, more complex queries tend to have longer completeness assessment time, with the case `w1d2` having the lowest and the case `w2d3` having the highest assessment time. Furthermore, query evaluation takes around 200 ms on average for all the cases. Hence, the completeness assessment time is comparable to the query evaluation time, and is even faster for simpler cases.

6 Conclusions and Future Work

Wikidata as a large-scale entity-centric KB might contain complete data that is currently still unexplored. We argue that for entity-centric, crowdsourcing KBs, SP-statements offer a good trade-off between expressiveness and practical usability of completeness information. In this paper, we have explored several ways to provide completeness information (in the form of SP-statements) for Wikidata. We have described COOL-WD, the first tool to manage completeness of a large knowledge base, which currently stores around 10,000 real completeness statements. COOL-WD allows one not only to produce but also to consume completeness statements: data completion tracking, completeness analytics, and query completeness assessment. COOL-WD offers great opportunities for further developing a high-quality linked open data space. Though in this paper we take Wikidata as a use case of completeness management and consumption, our ideas can also be adopted relatively easily to other entity-centric KBs.

An open, practical issue is the semantics of completeness for less well-defined predicates such as “medical condition” or “educated at,” as detailed in [15]. When it is unclear what counts as a fact for a predicate, it is also not obvious how to assert completeness. A possible solution is to devise a consensus or guidelines on what it means by a (complete) property, for instance: IMDb guidelines on complete cast or crew at <https://contribute.imdb.com/updates/guide/complete>. Further, the subjectivity of completeness along with potential impacts of COOL-WD has been discussed by the Wikidata community at <https://lists.wikimedia.org/pipermail/wikidata/2016-March/008319.html>.

Acknowledgment

This work has been partially supported by the projects “MAGIC”, funded by the province of Bozen-Bolzano, and “TQTK” and “CANDy”, funded by the Free University of Bozen-Bolzano. We would thank Amantia Pano and Konrad Hofer for their technical support for COOL-WD server, as well as Lydia Pintscher for pointers regarding Wikidata gadget development.

References

1. M. Acosta, E. Simperl, F. Flöck, and M. Vidal. HARE: A hybrid SPARQL engine to enhance query answers via crowdsourcing. In *K-CAP*, 2015.

2. X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *ACM SIGMOD*, 2015.
3. F. Darari, W. Nutt, G. Pirrò, and S. Razniewski. Completeness statements about RDF data sources and their use for query answering. In *ISWC*, 2013.
4. F. Darari, R. E. Prasojo, and W. Nutt. CORNER: A completeness reasoner for SPARQL queries over RDF data sources. In *ESWC Demos*, 2014.
5. F. Darari, S. Razniewski, R. E. Prasojo, and W. Nutt. Enabling fine-grained RDF data completeness assessment. In *ICWE*, 2016.
6. L. Galárraga, S. Razniewski, A. Amarilli, and F. M. Suchanek. Predicting completeness in knowledge bases. *Technical report*, 2016. Available at <http://a3nm.net/publications/galarraga2016predicting.pdf>.
7. P. J. Hayes and P. F. Patel-Schneider, editors. *RDF 1.1 Semantics*. W3C Recommendation, 25 February 2014.
8. A. Y. Levy. Obtaining complete answers from incomplete databases. In *VLDB*, 1996.
9. P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data quality assessment and fusion. In *Joint EDBT/ICDT Workshops*, 2012.
10. P. Mirza, S. Razniewski, and W. Nutt. Expanding Wikidata’s Parenthood Information by 178%, or How to Mine Relation Cardinalities. In *ISWC Posters & Demos*, 2016.
11. A. Motro. Integrity = validity + completeness. *ACM TODS*, 14(4):480–502, 1989.
12. H. Paulheim and C. Bizer. Improving the Quality of Linked Data Using Statistical Distributions. *Int. J. Semantic Web Inf. Syst.*, 10(2):63–86, 2014.
13. D. Rao, P. McNamee, and M. Dredze. Entity Linking: Finding Extracted Entities in a Knowledge Base. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 93–115. Springer, 2013.
14. S. Razniewski and W. Nutt. Completeness of queries over incomplete databases. *PVLDB*, 4(11):749–760, 2011.
15. S. Razniewski, F. M. Suchanek, and W. Nutt. But what do we actually know? In *AKBC Workshop at NAACL*, 2016.
16. M. Saleem, M. I. Ali, A. Hogan, Q. Mehmood, and A. N. Ngomo. LSQ: The Linked SPARQL Queries Dataset. In *ISWC*, 2015.
17. O. Savkovic, P. Mirza, S. Paramonov, and W. Nutt. MAGIK: managing completeness of data. In *CIKM Demos*, 2012.
18. Z. Sheng, X. Wang, H. Shi, and Z. Feng. Checking and handling inconsistency of DBpedia. In *WISM*. 2012.
19. G. Töpper, M. Knuth, and H. Sack. DBpedia ontology enrichment for inconsistency detection. In *I-SEMANTICS*, 2012.
20. D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
21. R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *J. of Management Information Systems*, 12(4):5–33, 1996.
22. A. Zaveri, D. Kontokostas, M. A. Sherif, L. Bühmann, M. Morsey, S. Auer, and J. Lehmann. User-driven quality evaluation of DBpedia. In *I-SEMANTICS*, 2013.