Data-driven Joint Debugging of the DBpedia Mappings and Ontology

Towards Addressing the Causes instead of the Symptoms of Data Quality in DBpedia

Heiko Paulheim

Data and Web Science Group, University of Mannheim, Germany
heiko@informatik.uni-mannheim.de

Abstract. DBpedia is a large-scale, cross-domain knowledge graph extracted from Wikipedia. For the extraction, crowd-sourced mappings from Wikipedia infoboxes to the DBpedia ontology are utilized. In this process, different problems may arise: users may create wrong and/or inconsistent mappings, use the ontology in an unforeseen way, or change the ontology without considering all possible consequences. In this paper, we present a data-driven approach to discover problems in mappings as well as in the ontology and its usage in a joint, data-driven process. We show both quantitative and qualitative results about the problems identified, and derive proposals for altering mappings and refactoring the DBpedia ontology.

Keywords: Knowledge Graph Construction, Knowledge Graph Debugging, Ontology Debugging, Data Quality, Data-driven Approaches, DBpedia

1 Introduction

Knowledge graphs on the Web are a backbone of many information systems that require access to structured knowledge, be it domain-specific or domain-independent [17]. The idea of feeding intelligent systems and agents with general, formalized knowledge of the world dates back to classic Artificial Intelligence research in the 1980s [21]. More recently, with the advent of Linked Open Data [3] sources like DBpedia [14] or YAGO [24], and by Google's announcement of the Google Knowledge Graph in 2012[1], representations of general world knowledge as graphs have drawn a lot of attention again.

In the Linked Open Data cloud, the DBpedia knowledge graph has become a central hub and widely used resource [22]. DBpedia is created from Wikipedia by harvesting information from infoboxes in Wikipedia pages. Those are mapped to an ontology in a community effort. Using those mappings, an A-box for the

[1] http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html

ontology T-box is automatically extracted using the DBpedia extraction framework [14]. While this approach allows for a reasonable coverage, there are various steps where errors can be introduced: for example, users may create wrong mappings or use the ontology in an unforeseen way. Those errors may lead to wrong extractions, which may limit the utility of the knowledge graph.

To address those shortcomings, various methods for *knowledge graph refinement* have been proposed. In many cases, those methods are developed by researchers outside the organizations or communities which *create* the knowledge graphs. They rather take the extracted DBpedia A-box and try to increase its quality by various means [17]. However, most of those approaches which target error correction focus on identifying single wrong assertions in the knowledge graph as a post-processing step to the construction. This means that the outcome is usually a long list of problematic assertions, which, depending on how reliable the approach is and how defensive the removal of assertions from the graph should be, need to be checked manually, which is a time-consuming process. Furthermore, upon a regeneration of the knowledge graph (e.g., with an updated set of heuristics and/or input corpus), the process needs to be run again. As new DBpedia releases are usually created on a bi-yearly basis[2], this means that the output of those approaches cannot be easily reused, and hence is discarded after six months in most of the cases.[3]

Therefore, a more sustainable way of handling data quality in DBpedia is required. In this paper, we propose a data-driven process which enriches the DBpedia knowledge graph with provenance information that allows for tracking the mapping which was responsible for creating an assertion in the DBpedia knowledge graph, or the ontology assertion that caused a statement to be inconsistent. By automatically identifying clusters inconsistent statements and corresponding mapping assertions, we are able to rank and pinpoint wrong mappings, as well as issues in the DBpedia ontology and its usage. By this, we can identify mappings to be changed, as well as proposals for refactoring the DBpedia ontology. Thereby, we step from identifying the *symptoms* of data quality in DBpedia to addressing their *causes*.

The rest of this paper is structured as follows. Section 2 discusses related work. In section 3, we sketch our approach, followed by a quantitative and qualitative analysis of the findings in section 4. We close with a summary and an outlook on future work.

2 Related Work

In this paper, we target the identification of *systematic errors* in the construction of the large-scale knowledge graph DBpedia.

There is a larger body of work which targets at finding errors in web knowledge graphs such as DBpedia. The approaches vary both with respect to the

[2] `http://wiki.dbpedia.org/why-is-dbpedia-so-important`

[3] A continuously updated knowledge graph, like DBpedia Live [10], generates a whole new set of challenges, which are out of scope of this paper.

methods employed as well as to the targeted type of assertions – i.e., identifying wrong type assertions, relational assertions, literals, etc. Methods found in the literature range from statistical methods [18] and outlier detection [6, 16, 27] to using external sources of knowledge, such as web search engines [13]. In addition, crowdsourcing [1] and games with a purpose [26] have been proposed as non-automatic means for identifying errors in knowledge bases. While such approaches can lead to a high precision, their main problem lies in scalability to larger knowledge bases [18].

Since many (but not all) wrong statements in a knowledge graph may surface as an inconsistency w.r.t. the underyling ontology, a few approaches rely on the use of reasoning given the knowledge graph's ontology for detecting inconsistencies. However, the DBpedia ontology – as many schemas used for providing Linked Open Data – is not very expressive, in particular with respect to the presence of disjointness axioms. Thus, there is a natural limitation for reasoning-based approaches. Hence, such approaches are often combined with ontology learning as a preprocessing step to enrich the ontology at hand [12, 15, 25], or exploit upper level ontologies [11, 19, 23]. Like the work presented in this paper, the latter two try to identify root causes: the former use T-box level reasoning to identify unsatisfiable concepts, the latter performs clustering on the reasoner's outcome to identify clusters of similar inconsistencies, which can often be attributed to a common root cause.

Reasoning on large-scale knowledge graphs, however, is a resource-intensive problem [20]. An approximation to the problem has thus been proposed in [4], in which schema-level reasoning on the DBpedia mappings and the DBpedia ontology is used to find inconsistent mapping assertions. Thus, the computational problem of dealing with a massive A-box is circumvented, while the T-box used in the reasoner is by several orders of magnitude smaller. In contrast to the work presented in this paper, this works at much faster runtimes, but cannot detect certain types of defects (e.g., the range of object properties is not respected).

3 Approach

Wrong assertions in a knowledge graph like DBpedia often surface as an inconsistency with the underlying ontology[4]. Therefore, in our approach, we first determine whether a single relation assertion is consistent with the subject's and object's types. Furthermore, we identify the DBpedia mappings that are responsible for the assertion at hand, and group the inconsistencies by the mapping. For each mapping, we can then compute scores which determine how frequently the mapping is involved in inconsistent statements, and hence, identify mappings which should be inspected by an expert. The inspection often reveals that either the mapping as such or the definition of the ontology concept it maps to are problematic.

[4] However, not all wrong assertions lead to inconsistencies, and not all inconsistencies are due to wrong assertions.

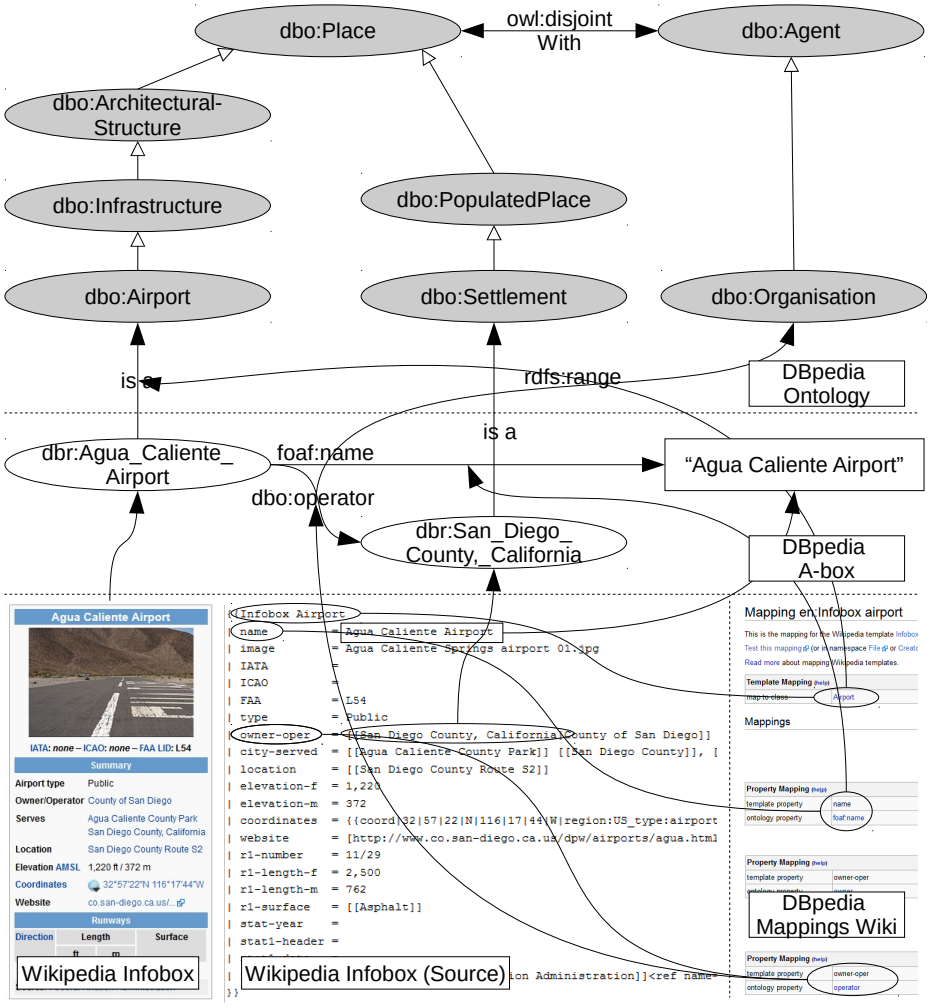

Fig. 1: Example of the Extraction from DBpedia. Infoboxes in Wikipedia (lower left) are mapped to the DBpedia ontology in the DBpedia Mappings Wiki (lower right). With the help of those mappings, entities and assertions are extracted, the DBpedia A-box (middle), which use the DBpedia ontology (top).

3.1 Preliminaries

DBpedia is extracted from Wikipedia infoboxes. In the DBpedia Mappings Wiki[5], infoboxes are mapped to classes in the DBpedia ontology, and infobox keys are mapped to properties in the DBpedia ontology. These mappings are created in a community-driven process.

[5] `http://mappings.dbpedia.org`

The DBpedia extraction code uses those mappings to extract entities and assertions which form the A-box of DBpedia. Fig. 1 illustrates the process. It shows the infobox from the Wikipedia page of the Agua Caliente Airport[6], together with its source code[7], which shows the use of an airport infobox. In the DBpedia Mappings Wiki, this infobox is mapped to the ontology type `dbo:Airport`[8], and its infobox keys are mapped to properties in the DBpedia ontology. The figure depicts two of those mappings: the infobox key `name` is mapped to `foaf:name`, and the infobox key `owner-oper` is mapped to `dbo:operator`.

With the help of those mappings, instances and assertions are created. In the example, there is a new resource created for the Wikipedia page at hand, i.e., `dbr:Agua_Caliente_Airport`, as well as for the Wikipedia page linked to in the infobox entry for `owner-oper`, i.e., `dbr:San_Diego_County,_California`. The resulting assertions are:

```
dbr:Agua_Caliente_Airport rdf:type dbo:Airport .
dbr:Agua_Caliente_Airport foaf:name "Agua Caliente Airport"@en .
dbr:Agua_Caliente_Airport dbo:operator
                          dbr:San_Diego_County,_California .
```

Furthermore, from the linked Wikipedia page for San Diego County, the following axiom is created, using the same mechanism for a different infobox:

```
dbr:San_Diego_County,_California a dbo:Settlement .
```

Repeating this process for each page in Wikipedia leads to the DBpedia A-box, which consists of millions of entities and assertions.

3.2 Datasets Used

Similar to our work proposed in [19], we use DBpedia together with the DOLCE-Zero ontology [7, 8] in order to be able to add more top level disjointness axioms and hence discover more inconsistencies. We use the most recent DBpedia 2016-04 release with the corresponding ontology.

However, the mapping from the DBpedia 2016-04 ontology to DOLCE has an issue with the mappings which have originally been defined as `rdfs:subclassOf` relations, but been changed to `owl:equivalentClass` mappings in the current release.[9] This leads to certain problems, as this example illustrates:

---

[6] `https://en.wikipedia.org/wiki/Agua_Caliente_Airport`, retrieved on December 6th, 2016

[7] `https://en.wikipedia.org/w/index.php?title=Agua_Caliente_Airport&action=edit`, retrieved on December 6th, 2016

[8] Throughout this paper, we use the following namespace conventions: `dbo=http://dbpedia.org/ontology/`, `dbr=http://dbpedia.org/resource/`, `foaf=http://xmlns.com/foaf/0.1/`, `rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#`, `rdfs=http://www.w3.org/2000/01/rdf-schema#`

[9] See `https://sourceforge.net/p/dbpedia/mailman/message/35452137/`

```
dbo:GovernmentType owl:equivalentClass dul:Concept .
dbo:MusicGenre owl:equivalentClass dul:Concept .
```

From this, a reasoner would conclude that `dbo:GovernmentType owl:equivalentClass dbo:MusicGenre` also holds. The class `dbo:MusicGenre` thereby becomes insatisfiable. In order to work around these issues, we have replaced all `owl:equivalentClass` mappings between the DBpedia and DOLCE-Zero ontology with `rdfs:subclassOf`.[10]

3.3 Identifying and Grouping Inconsistencies

An inconsistency arises if an assertion (or a set of assertions) is incompatible with the ontology that defines constraints on those assertions. In the example in Fig. 1, the range of the relation `dbo:operator` is restricted to the class `dbo:Organisation`. However, the object of the relation at hand is `dbr:San_Diego_County,_California`, which is an instance of `dbr:Settlement`.

From the ontology, we can see that `dbo:Agent` (which is a superclass of `dbo:Organisation`) and `dbo:Place` (which is a superclass of `dbo:Settlement`) are disjoint. Hence, the relation assertion, together with the object's type, forms an inconsistency.

For detecting such inconsistencies, we create minimal A-boxes, consisting of a relation assertion and its subject's and object's types. Those are then loaded into the HermiT reasoner [9], together with the ontologies. The reasoner computes whether the A-box is consistent or not. Furthermore, the proof tree the reasoner provides contains all statements that together form the inconsistency.[11] In the example in Fig. 1, those consist of the object's type assertion, the relation assertion, the range statement, as well as chain of subclass statements for `dbo:Organisation` and `dbo:Settlement` up to `dbo:Agent` and `dbo:Place`, and the disjointness axiom between the latter two.

Since we are interested in identifying wrong mapping assertions as well as problems within the ontology, we identify the mapping statements that contribute to an inconsistency. However, in DBpedia, there is no provenance information which tracks which mapping element was used to create which assertion. Hence, we first need to reconstruct that provenance information. For this reconstruction, we use two sources:

1. The set of templates which are used on subject's and object's original Wikipedia pages. That information is provided with the DBpedia release.
2. The mappings from the DBpedia Mappings Wiki translated to RDF using the RML vocabulary. [4, 5]

In the first step, we identify those templates which are used on the subject's and object's original Wikipedia page. In our running example, that is `Infobox`

[10] Note that this change only weakens the original assertions, thus, it cannot introduce additional inconsistencies.

[11] There can be multiple proof trees for the same inconsistency. However, in our approach, we only pick one at random.

```
Given an assertion a (S r O)
Get T(S) and T(O) // templates used in the assertion's subject's and object's Wikipedia page
MapR(r) = Mapping elements in T(S) that assign a relation r
For all asserted types T of S
        MapT(S) = Mapping elements in T(S) that assign a type from T
For all asserted types of T of O
        MapT(O) = Mapping elements in T(O) that assign a type from T

Use reasoner to compute consistency of a
If a is inconsistent
        If explanation contains a
                Mark all entries in MapR(r)
        If explanation contains subject type assertion with type T
                Mark all entries in MapT(S)
        If explanation conatins object type assertion with type T
                Mark all entries in MapT(O)

For all marked mapping elements
        increase inconsistency counter for element
For all non−marked mapping elements
        increase consistency counter for element
```

Fig. 2: Pseudocode for identifying mapping elements contributing to inconsistencies

`airport` for the subject, and `Infobox settlement`, `Authority control`, and `See also` for the object.

In a second step, we retrieve all mapping elements for those templates. From that subset, we identify those which are used to assert the types and the relation at hand. In our example, this would be the mapping elements that assert the class `dbo:Airport` for `Infobox airport` (1), the relation `dbo:organization` for the infobox key `owner-oper` (2), and the type `dbo:Settlement` for the `Infobox settlement` (3).

When this identification is done, we run the reasoner to compute the inconsistency, and, if the statement is inconsistent, the explanation. Using that explanation, we mark all the mapping elements that are responsible for a statement used in the explanation. With those marks, we maintain two counters for each mapping element m: how often it was generating an assertion involved in an inconsistency (inconsistency counter i_m), and how often it was not (consistency counter c_m). Figure 2 illustrates this approach.

In our example, i_m is increased for (2) and (3), i.e., the mapping elements responsible for the relation assertion and the object type assertion,while c_m is increased for (1), i.e., the mapping element responsible for the subject type assertion, is increased, since the type assertion for the subject was not involved in the inconsistency.

3.4 Scoring Inconsistencies

Given the consistency and inconsistency counts for each mapping element, we can compute a number of scores for each mapping element. Given the hypothesis that a mapping element m is problematic, and the two counts for the statements produced by m that were involved in explanations for inconsistencies (i_m) and those that were not (c_m), we first use two metrics borrowed from association rule mining, i.e., support and confidence [2]:

$$s(m) := \frac{i_m}{N} \tag{1}$$

$$c(m) := \frac{i_m}{i_m + c_m} \tag{2}$$

Here, N is the total number of statements in the knowledge base. As we are looking only at relation assertions, N is roughly 17.6 million. In turn, this means that while the confidence can easily grow towards 1 (e.g., if a wrong mapping element produces mostly wrong statements leading to inconsistencies), this is not true for support: even a mapping element leading to 100,000 inconsistent cases (which is a mapping element which we would want to achieve a high score) would have a support value of only 0.005. That makes it difficult to compute a common score from the two, since, although they both theoretically produce a score in the $[0; 1]$ interval, the actual scores practically come in different scales. Hence, to overcome this problem, we propose to use *logarithmic support*, defined as

$$logs(m) := \frac{log(i_m + 1)}{log(N + 1)} \tag{3}$$

In contrast to standard support, logarithmic support works in orders of magnitude, i.e., a log support of 0.5 means that the order of magnitude of the counted incoherences affects half the order of magnitude of all the axioms. Thus, a mapping element leading to 100,000 inconsistent cases, as discussed above, would achieve a fairly high logarithmic support score of 0.69.

Finally, to assign ratings to the mapping elements and pick the most promising cases for inspection by an expert, we want to consider mapping elements which achieve both a high (logarithmic) support, i.e., that lead to a significant number of inconsistencies, and that have a high confidence, i.e., those inconsistencies are not mere noise. Hence, we use the harmonic mean of logarithmic support and confidence as a final rating score for statements:

$$score(m) := \frac{2 \cdot logs(m) \cdot c(m)}{logs(m) + c(m)} \tag{4}$$

For our experiments, we computed mean (μ) and standard deviation (σ) of s, logs, and c. We observe $\mu_s = 0.0002$, $\sigma_s = 0.003$, $\mu_{logs} = 0.179$, $\sigma_{logs} = 0.139$, $\mu_c = 0.114$, and $\sigma_c = 0.260$. This shows that the distribution of logs and c actually resemble one another, and they can thus be safely combined using the harmonic mean.

Using that rating function, we can group the output of the identified inconsistencies by mapping elements, assing a score to each mapping element, and inspect the high scoring elements for an identification of common problems in the construction process of DBpedia.

Note that in our running example, we had flagged two mapping elements: one mapping the type `dbo:Settlement` to `Infobox settlement`, and one mapping the infobox key `owner-oper` to `dbo:operator`. The property mapping has a confidence of 0.153 and a logarithmic support of 0.377, leading to a score of 0.218. In contrast, the type mapping for the object has only a confidence of 0.0004, at a similar logarithmic support of 0.344, leading to a score of 0.0008. Thus, in this case, the likelihood that the statements extracted from the property mapping are involved in an inconsistency is much higher than the likelihood of the object type mapping. Hence, given a suitable lower bound for the overall score, we would examine the case only once. Here, an expert would diagnose that the range assertion of `dbo:operator` is not compatible with a larger fraction of the assertions using the property.

4 Findings

In this section, we report about the number of problems identified, as well as present typical problems and proposed solutions.

4.1 Quantitative Results

In total, there are 63,981 mapping elements in the snapshot of the RML translation of the DBpedia Mappings Wiki we used in our experiments.[12] Out of those, 3,454 are identified to produce a statement which is in one of the A-boxes we inspect.[13] We identified a total number 1,117 (i.e., 32.3%) of mapping statements involved in at least one inconsistency.

Fig. 3 depicts the distribution of the mapping elements that produce at least one statement involved in an inconsistency. We can observe that there is a larger number of mapping elements with confidence 1, i.e., all the statements they produce are involved in inconsistencies. Furthermore, there is a larger cluster of mapping elements with a confidence below 0.05, i.e., we can assume that the inconsistencies are produced by other effects than the mapping element or the ontology (e.g., wrong statements in Wikipedia, incorrect extraction of URIs from hashed URLs, etc. – see, e.g., [19] for a discussion).

We mainly inspected those mapping elements which receive a high overall score as defined in (4), i.e., those that are depicted in the top right corner of the scatter plot. From those, we derived a set of typical problems.

[12] `http://rml.io/data/DBpediaAll.rml.nt`, retrieved on November 2nd, 2016

[13] This is indeed an interesting discrepancy. A larger number of mapping elements produces literal-valued statements, which are out of scope of our inspection. Furthermore, there might be outdated mappings for infoboxes no longer in use.

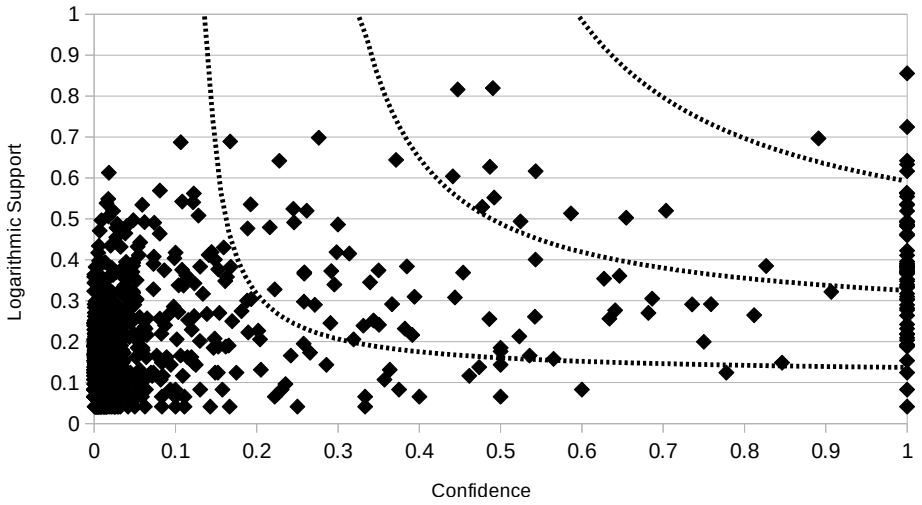

Fig. 3: Distribution of flagged mapping elements according to logarithmic support and confidence. The three dashed lines depict the isometrics of the overall score of 0.25, 0.5, and 0.75.

4.2 Mapping Errors

Pure erroneous mappings are scarce. In most cases, there is a mapping to a property which seems correct, but the property has a different meaning defined in the ontology, as can be seen by inspecting the domain and range of the property. When inspecting the distributions of the actual usage of the property, however, it becomes evident its intended semantics is often different from how it is understood and used by the crowd contribution to the DBpedia Mappings Wiki.

Mapping to Wrong Property In the simplest case, an infobox key is mapped to a wrong property in the DBpedia ontology, where a correct one actually exists.

One example for a wrong mapping is the `branch` key in the `Military unit` infobox, which is mapped to `dbo:militaryBranch`. However, that property expects a person in the subject position. The correct property would be `dbo:commandStructure`. This affects 12,172 assertions in DBpedia (i.e., 31% of all assertions of `dbo:militaryBranch`).

Another interesting example is the mapping of depictions in various infoboxes (e.g., music covers) to `dbo:picture`. The extraction code does not generate a link to the media object as such, but tries to parse the image file name into a DBpedia resource. Thus, the 3,354 assertions using that property are mostly inconsistent, covering mostly places (2,021, or 64.5%) and persons (465, or 23.0%), . This is one of the rare cases where a mapping should not be altered, but discarded altogether. Some instantiations of that problem are the statements

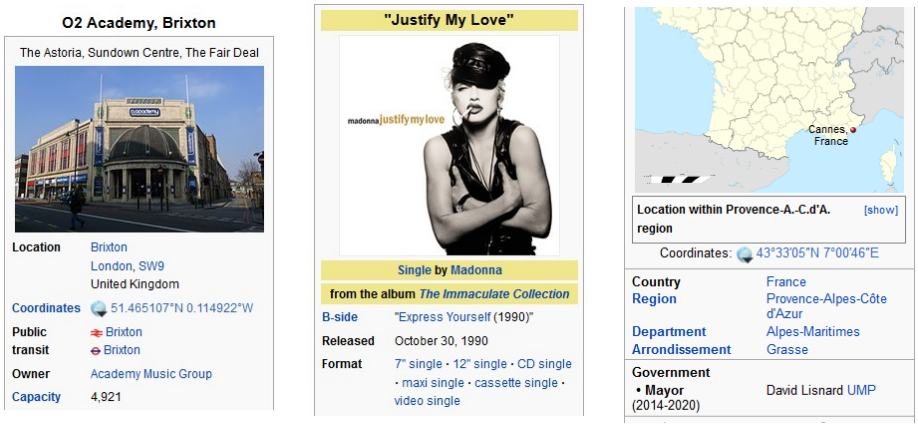

Fig. 4: Three example infoboxes (from the Wikipedia pages *Justify My Love*, *Brixton Academy*, and *Cannes*)

```
dbo:Brixton_Academy dbo:picture dbo:Brixton .
dbo:Justify_My_Love dbo:picture dbo:Madonna_(entertainer) .
```

The corresponding two infobox snippets are depicted in Fig. 4 (left and middle).

A similar example is the mapping of the property `dbo:mayor`. For a majority of instances of that property (436, or 47.3%), the corresponding party is extracted as an object of the property, while a considerably smaller fraction (234, or 25.4%) are actually persons. A corresponding infobox is depicted on the right hand side in Fig. 4.

Missing Properties Similarly to mappings to wrong properties, some infobox keys are mapped to an inadequate ontology property, but a correct one cannot be found in the ontology. Here, the ontology has to be enhanced.

The property `dbo:president` is designed to assert that a person is the president of an organization (e.g., a state or a company). However, this is only the minority of the cases (2,600 assertions, or 23.7%). The vast majority of the subjects is of type `dbo:Person` (8,354, or 76.2%). Here, the property is used to assert that a person (e.g., a minister) was serving *for* a certain president. Here, the introduction of a new property (e.g., `dbo:servedFor`) should be considered.

Another example is the mapping of the infobox key `instruments` in the `Infobox music genre` to `dbo:instrument`, whose domain is `dbo:Artist` (i.e., the property is intended to relate a artist to the instrument (s)he plays). This affects 707 assertions in total. Again, the introduction of a new property (e.g., `dbo:characteristicInstrument`) should be considered.

Further examples include the use of `dbo:species` for fictional characters in movies or books, while it is intended (and most widely used) for defining

biological taxonomies, or `dbo:director`, which is meant to be the director of a movie, but also used to denote, e.g., the director of a festival or other event.

4.3 Problems in the Ontology

While some errors can be tracked down to individual mappings, others keep recurring for many infobox types. Hence, their root is usually not the mappings as such, but problematic definitions in the ontology. One example is the `dbo:operator` property, which has been used as a running example above. While having `Organisation` as its defined range, out of 13,425 objects of `dbo:operator`, 9,529 are of that type, while 1,092 have populated places (i.e., cities, counties, etc.) being their operator. This holds for many classes, such as airports, libraries, or stadiums. In that case, the range of `dbo:operator` should either be broadened, or cities, counties etc. should be a subclass of both `dbo:Place` and `dbo:Agent` in order to allow for the observed polysemy (cities, at least as they are used in DBpedia, are both a geographic and a social object).

A similar case is the range of properties like `dbo:architect`, `dbo:designer`, `dbo:engineer`, etc. Those often define a `dbo:Person` in the range, but in many cases, there are companies in the object position: `dbo:architect` has 1,480 (8.6%) organizations and 8,538 (49.5%) persons, designer has 806 organizations (7.8%) and 5,298 (50,4%) persons, and `dbo:engineer` has even a majority 153 (58.4%) organizations and 32 (12.2%) persons. Here, the range of those properties should be broadened to `dbo:Agent` for covering both persons and organizations.

A large-scale problematic mapping is the mapping of the property `dbo:team`, which is mainly used for subjects of type `dbo:CareerStation` (351,580 out of 1,287,645, or 27.3%)[14], but also for `dbo:Person` (160,452 out of 1,287,645, or 12.5%). The two are clearly not compatible, and since the corresponding super property in the DOLCE-Zero (`dul:isSettingFor`) expects a `dul:Situation` in the subject position, the latter cases lead to inconsistencies. Here, a more uniform usage of the property and hence, a more consistent modeling of athletes and the team(s) they belong to over time, should be enforced.

Another large-scale issue in the ontology is the use of bands (i.e., `dbo:Band`) and musical artists (i.e., `dbo:MusicalArtist`) together with properties that expect either one or the other (e.g., `dbo:associatedBand`. However, the corresponding objects are more or less equally distributed across both classes. In fact, the corresponding infobox keys, which do not distinguish between associated musical artists and bands, are currently mapped to *both* `dbo:associatedBand` and `dbo:associatedMusicalArtist`. Here, we propose the refactoring into a common property `dbo:associatedMusicAct`, whose range is the union of `dbo:Band` and `dbo:MusicalArtist`.

4.4 Problems with the Mapping to DOLCE-Zero

The mapping to DOLCE-Zero provides additional disjointness axioms which help discovering more inconsistencies. However, in a few cases, a property is

[14] A large majority of the subjects is not typed at all.

used for a certain purpose in the majority of cases, which is incompatible with the formalization DOLCE.

One such example is the property `dbo:commander`, although not defining an explicit domain in the DBpedia ontology, is a subproperty of `dul:coparticipatesWith`, which has the domain `dul:Object` by inference using the DOLCE-Zero ontology, which is disjoint with `dul:Event`. However, the majority (i.e., 11,831 out of 12,841, or 92.1%) of all subjects using this property are `dbo:MilitaryConflicts`, a subclass of `dul:Event`. Those are mainly created by a mapping in the `Military conflict` infobox.

5 Conclusion and Outlook

In this paper, we have introduced a data-driven approach targeted at increasing the data quality in DBpedia. The proposed approach searches for inconsistencies with the underlying ontology, also leveraging the linked top level ontology DOLCE-Zero, and tries to identify common root causes of those inconsistencies.

With our approach, we were able to identify quite a few problems of different kinds. We found pure mapping errors, hints for missing properties, as well as problematic domain and range restrictions in the DBpedia ontology.

To leverage the results, we have started fixing the problematic mappings, where appropriate, so that the next version of DBpedia will not contain the problematic assertions anymore. In cases where changes to the ontology are required, be it domain/range changes or the introduction of new properties, discussion threads have been started, since some of the changes may have long-reaching consequences and should thus be considered carefully.

For the moment, an expert has to review the problems identified in our approach, and manually classify them as wrong mappings, problems in the ontology, etc. In future work, we aim at developing a set of data-driven heuristics which perform these classifications automatically. As a long term high level vision, we foresee the creation of a knowledge graph validator which, provided with a knowledge graph and statement-level provenance information (i.e., what were the mechanisms that led to the inclusion of a particular assertion in the knowledge graph), issues fine-grained qualified suggestions on how to revise the creation process for increasing the knowledge graph's data quality. Tooling-wise, this could be implemented in an automatic reporting system, or even in the mappings Wiki for issuing live warnings upon editing time.

The work presented in this paper is a first step in this direction. While most of the approaches for increasing the data quality in knowledge graphs discussed in the literature so far are pure post-processing approaches which aim at the identification, elimination, or correction of problematic statements, the work presented in this paper is one of the rare examples that go one step beyond by identifying the root causes and fixing the cause of the data quality problem instead of remedying the symptoms.

Acknowledgements

The author would like to thank the numerous people involved in the DBpedia project for their past, ongoing, and future efforts, as well as the authors of [4] for providing the DBpedia mappings in RML.

References

1. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S., Lehmann, J.: Crowdsourcing Linked Data quality assessment. In: The Semantic Web–ISWC 2013, LNCS, vol. 8219, pp. 260–276. Springer, Berlin Heidelberg (2013)
2. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International journal on semantic web and information systems 5(3), 1–22 (2009)
4. Dimou, A., Kontokostas, D., Freudenberg, M., Verborgh, R., Lehmann, J., Mannens, E., Hellmann, S.: DBpedia Mappings Quality Assessment. In: International Semantic Web Conference – Posters and Demonstrations (2016)
5. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: Rml: A generic language for integrated rdf mappings of heterogeneous data. In: LDOW (2014)
6. Fleischhacker, D., Paulheim, H., Bryl, V., Völker, J., Bizer, C.: Detecting Errors in Numerical Linked Data Using Cross-Checked Outlier Detection. In: The Semantic Web–ISWC 2014, LNCS, vol. 8796, pp. 357–372. Springer, Switzerland (2014)
7. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A.: Sweetening WordNet with DOLCE. AI Magazine 24(3), 13–24 (2003)
8. Gangemi, A., Mika, P.: Understanding the semantic web through descriptions and situations. In: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, pp. 689–706. Springer (2003)
9. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an owl 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
10. Hellmann, S., Stadler, C., Lehmann, J., Auer, S.: Dbpedia live extraction. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 1209–1223. Springer (2009)
11. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked Data Is Merely More Data. In: AAAI Spring Symposium: linked data meets artificial intelligence. vol. 11 (2010)
12. Lehmann, J., Bühmann, L.: ORE – a tool for repairing and enriching knowledge bases. In: The Semantic Web–ISWC 2010, LNCS, vol. 6497, pp. 177–193. Springer, Berlin Heidelberg (2010)
13. Lehmann, J., Gerber, D., Morsey, M., Ngomo, A.C.N.: DeFacto – Deep Fact Validation. In: The Semantic Web–ISWC 2012, LNCS, vol. 7649, pp. 312–327. Springer, Berlin Heidelberg (2012)
14. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web Journal 6(2) (2013)

15. Ma, Y., Gao, H., Wu, T., Qi, G.: Learning Disjointness Axioms With Association Rule Mining and Its Application to Inconsistency Detection of Linked Data. In: Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., Pan, J.Z. (eds.) The Semantic Web and Web Science, Communications in Computer and Information Science, vol. 480, pp. 29–41. Springer, Berlin Heidelberg (2014)
16. Paulheim, H.: Identifying Wrong Links between Datasets by Multi-dimensional Outlier Detection. In: International Workshop on Debugging Ontologies and Ontology Mappings. CEUR Workshop Proceedings, vol. 1162, pp. 27–38 (2014)
17. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web 8(3), 489–508 (2017)
18. Paulheim, H., Bizer, C.: Improving the Quality of Linked Data Using Statistical Distributions. International Journal on Semantic Web and Information Systems (IJSWIS) 10(2), 63–86 (2014)
19. Paulheim, H., Gangemi, A.: Serving DBpedia with DOLCE–More than Just Adding a Cherry on Top. In: International Semantic Web Conference. LNCS, vol. 9366. Springer, International (2015)
20. Paulheim, H., Stuckenschmidt, H.: Fast approximate a-box consistency checking using machine learning. In: Extended Semantic Web Conference. pp. 135–150. Springer (2016)
21. Russell, S., Norvig, P.: Artificial Intelligence: a Modern Approach. Pearson, London (1995)
22. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the Linked Data Best Practices in Different Topical Domains. In: International Semantic Web Conference. LNCS, vol. 8796. Springer, International (2014)
23. Sheng, Z., Wang, X., Shi, H., Feng, Z.: Checking and Handling Inconsistency of DBpedia. In: WISM'12. pp. 480–488 (2012)
24. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In: 16th international conference on World Wide Web. pp. 697–706. ACM, New York (2007)
25. Töpper, G., Knuth, M., Sack, H.: DBpedia Ontology Enrichment for Inconsistency Detection. In: Proceedings of the 8th International Conference on Semantic Systems. pp. 33–40. ACM, New York (2012)
26. Waitelonis, J., Ludwig, N., Knuth, M., Sack, H.: WhoKnows? – Evaluating Linked Data Heuristics with a Quiz that Cleans Up DBpedia. International Journal of Interactive Technology and Smart Education 8(4), 236–248 (2011)
27. Wienand, D., Paulheim, H.: Detecting Incorrect Numerical Data in DBpedia. In: The Semantic Web: Trends and Challenges, LNCS, vol. 8465, pp. 504–518. Springer, International (2014)