Mechanism for inconsistency correction in the DBPedia Live

Túlio Brandão Soares Martins *Julio Cesar dos Reis*

Technical Report - IC-19-08 - Relatório Técnico
September - 2019 - Setembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Mechanism for inconsistency correction in the DBPedia Live

Túlio Brandão Soares Martins
Julio Cesar dos Reis
Institute of Computing, University of Campinas, São Paulo, Brazil

2019

Abstract

DBpedia is a huge resource available in the Web of Data. It is relevant to update this dataset based on new information appearing in the Wikipedia. However, this operation can provide inconsistencies in the dataset for chapters in different languages. Although existing literature has defined tools to update *DBpedia*, there is a lack of studies related to understand and detect multi-chapter inconsistencies in its evolution. In this work, we define a set of inconsistency classes related to triple changes in the *DBpedia* to inform a software tool suited to detect instances of these classes when new data is updated for secondary languages, removed or inserted in the dataset. We conduct an experimental evaluation to assess the types of inconsistencies in several versions of the dataset. We demonstrate that the identified inconsistencies appear in the evolution of DBpedia and propose a solution to correct these inconsistencies. Our evaluation assessing the proposed technique reveled its usefulness. Our results show that the proposed classes of inconsistencies combined with the defined solution can turn the *DBpedia* more reliable overtime.

1 Introduction

The continuous growth of the Web raises several questions on how to handle big chunks of information that are somewhat related. Questions of which the web of Data seeks to answer by finding ways to link new information added constantly. In an effort to centralize and structure data, the *DBpedia* project [1] was proposed to take information from Wikipedia and turn it machine readable.

The DBpedia has become a relevant tool that uses the massive amounts of data from Wikipedia, and can be very useful for other parties. Currently, projects use

DBpedia to describe and relate objects, such as the *DBpedia Mobile*, that describes locations and relates to other interesting information which can be very useful for travelers and people in general learning about places and cultures.

The content of Wikipedia naturally grows overtime. Simultaneously, *DBpedia* must keep reliable and up-to-date in multiple chapters described in different languages. Currently, *DBpedia* is updated in two ways: 1) through large dumps of data from Wikipedia which are altered periodically - these reformulate the database in a large scale, but are infrequent; 2) through *DBpedia Live* [8], as a software tool to update DBpedia RDF triples in real time. It relies on an Extraction Framework that collects content directly from Wikipedia in the moment a wiki page is altered.

Due to the key relevance of *DBpedia*, its RDF triple facts must provide reliable information in distinct cross-language datasets. However, in the update procedure, inconsistencies can be generated in the extraction process in other languages. For instance, some triples that exist in a dataset are not consistent with the ontology in place. In addition, many resources in other languages are too broadly defined and are not properly checked for inconsistency. These inconsistencies can make *DBpedia* unreliable for third parties to use.

The maintenance of the consistency in *DBpedia* plays a key role for sources posing queries to it. The results of query processing can be affected by inconsistent triples. Also cross-lingual information retrieval requires that information described in distinct language is formally consistent to avoid inadequate results.

The research on how to make *DBpedia* a reliable knowledge base is extensive and continuous. The current literature has approached inconsistency classification [5] [3] [11] and data co-evolution [9] [7] [4]. Some investigations have approached language specific inconsistencies [3]. However, to the best of our knowledge, it lacks studies applied to the live extraction in a way to filter inconsistencies in real time updates for distinct *DBpedia* language datasets.

In this investigation, we formally define a set of inconsistency classes that can occur during the *DBpedia* Live extraction. Based on implemented procedures for the inconsistency identification, we carry out an analysis to measure the frequency in which they appear on the *DBpedia* of various languages other than English. We argue why these inconsistencies are more frequent in other languages and demonstrate the possibility of filtering them during the execution of the DBpedia Live extraction. As a second contribution, we devise and implement a mechanism for inconsistency correction in *DBPedia Live*. Our conducted evaluation of inconsistency correction showed that the number of inconsistencies decreased via our mechanism and the extraction process was improved.

This article is organized as follows: Section 2 presents the background work. Section 3 defines the types of inconsistencies and conduct analyses in real update of *DB-Pedia* datasets. Section 4 proposes a solution to correct the inconsistencies and avoid their recurrence in addition to present the evaluation results. Afterwards, Section 5 discusses our obtained findings. Finally, Section 6 presents the final considerations.

2 Background

Whereas subsection 2.1 presents studies concerned to the consistency of RDF triples and in *DBPedia* triples, subsection 2.2 describes existing work addressing the evolution and update of RDF graphs.

2.1 Consistency of RDF data triples

Bizer & Heath [2] described the concepts of Linked Data and its principles. They situated the relevance of Linked Data and indicated DBPedia as a key example highlighting the important of keeping such data consistent overtime. Pern *et al.* [11] defined inconsistencies in RDF data and indicated ways of detecting such inconsistencies utilizing queries. Our investigation takes a similar approach in defining inconsistencies in RDF and extend it to other types of inconsistencies. Our approach employs a similar technique utilizing queries for the detection, but applied specifically to the *DBpedia Live* [8].

Cabrio *et al.* [3] proposed the classification of inconsistencies regarding different language databases. The authors defined the different types of the named "language inconsistencies" and mapped ways of correcting such inconsistencies. Our work uses that idea to find other types of inconsistencies in datasets described in different language with multiple references of one resource to others.

Topper *et al.* [5] defined inconsistencies and proposed to increment the current *DBpedia* ontology with the goal of improving the detection of inconsistencies. Their method consisted of a thorough analysis of each property's domain and range by applying specific modifications for each property. We utilized that idea in the proposition of updating specific definitions of the ontology, but most of the correction techniques proposed in our work requires small alteration of the DBpedia ontology structure. We utilized a similar method of detection through each individual property implemented in an extraction tool of the *DBpedia Live* framework.

Paulheim & Gangemi [6] proposed ontology enhancement with the addition of another ontology, *DOLCE*, to improve consistency in the *DBpedia.* On the contrary, our proposition defines a solution requiring minimal alterations to the current ontology to achieve a more consistent dataset overtime.

2.2 RDF graph updates

Literature has approached the problem of updating RDF graphs, addressing mostly the precautions that should be made, how they must obey ontology axioms and possible inconsistencies that can be caused through these updates. Endris *et al.* [7] explored how the local duplication of data can eventually cause inconsistencies in RDF graphs such as *DBpedia* during it's updates.

In addition, Konstandinidis *et al.* [4] approached the evolution of ontologies based on new information to maintain consistent RDF datasets. Utilizing specific algorithms to alter the rules of ontology for properties (such as domain and range), the authors proposed an advanced improvement in consistency. In our work, instead of redefining the properties we looked to better define the resources considering we were dealing specifically with DBpedia.

In relation to the co-evolution of RDF data, Faisal & Endris *et al.* [9] addressed the treatment of consistency issues given modifications such as updating and removal affecting a triple subject. This approach is relevant because it treats the evolution of a triple database.

Auer & Herre *et al.* [10] explored the evolution of RDF datasets and proposed an ontology evolution based on atomic changes including additions or deletions of information. By utilizing change operations and generating metainformation regarding changes in the ontology, their work aimed to better treat the consistency of RDF databases. Our investigation handles the consistency of triple adding specifically applying it to the DBpedia Extraction Manager for the *DBpedia Live*.

In an overall, existing work in literature show that there is considerable study to the detection of inconsistencies, their correction through ontology changes or addition of information to the RDF graphs. However, to the best of our knowledge, implemented approaches require several ontology alterations, which is unfeasible for a RDF dataset as large as *DBpedia*. Also, it is not sufficient to apply them directly to the *DBpedia Live* and it's extraction manager, which is performed in our work.

The platform *DBpedia Live* [8] aims to extract triples without requiring one single dump. As long as the Live platform is running, every Wikipedia page update is queued to later on be extracted into triples, which are inserted in the *DBpedia* dataset. This investigation worked with the live mirror[1], a local copy of the *DBpedia Live*, that is open source that simulates the extraction and update of *DBpedia*.

Figure 1 presents the workflow the extraction process in the *DBpedia Live*. Every update in a Wikipedia page is serialized in Wikimedia's OAI-PMH, which is a protocol for transmission of data and meta data in repositories. The *DBpedia Live* has access to a channel with this protocol and thus receives every serialized page update. The updated pages go into a queue to the API Source and are ready to be parsed. The parser separates the information and organizes it accordingly. It compares every update to the original page and verifies which information must be added. It then transfers each new information to a different extractor, *e.g.*, if a new geographic location is added to a page, this added information is sent to the geographic location

[1]https://github.com/dbpedia/dbpedia-live-mirror

extractor.

Our work focuses on the mapping extractor, which is the extractor utilized when the added information of resource makes a reference to another resource, and both are extracted to their equivalents in *DBpedia*. The extractor converts the information into RDF triples. The output of all of the extractors (the triples) is then united and arranged, and then serialized in the *DBpedia* dataset.

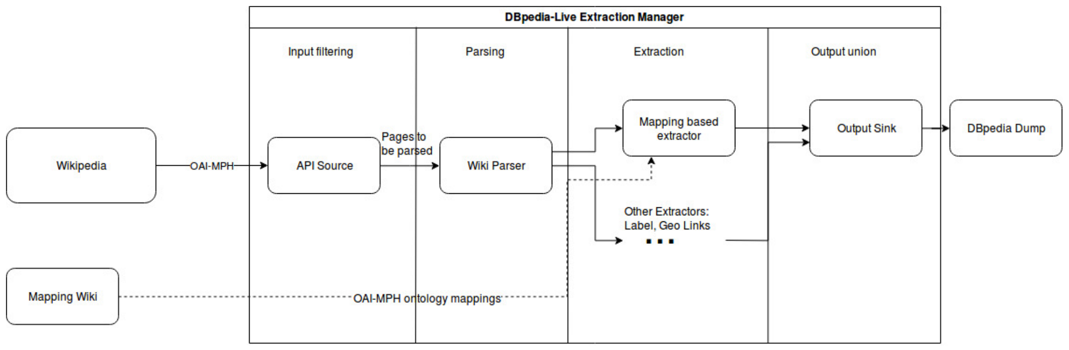

Figure 1: Workflow of the *DBpedia Live Extraction*.

Our investigation addresses inconsistency detection and classification in the context of the *DBpedia Live extraction manager*. The goal is to automatically handle and correct inconsistencies in the execution of the framework on-the-fly. Our contribution treats inconsistencies in *DBpedia* overtime and helps making this RDF dataset further reliable.

3 Identifying inconsistencies in DBPedia updates

Subsection 3.1 provides formal definitions related to the addressed types of inconsistencies. Subsection 3.2 presents our empirical study in analyzing the classification of the defined inconsistencies in the DBPedia update.

3.1 Formal definitions

RDF Triple - A *RDF triple* $t = (s, p, o)$ consists of three elements in which s is the subject of the triple, p is the predicate of the relation between s and o defined as a property; and o is known as the object of the triple. We define a RDF dataset as a set of triples, such that, $R = \{t_1, t_2, ..., t_n\}$.

Ontology - Ontology refers to a collection of concepts and axioms that the triples must obey. For every s subject, there are classes which they belong to. Given those classes, the p properties establish which type of relationship s can relate to the objects o. An ontology contains the vocabulary of all the classes and properties available for the relations. if the subject s belongs to a certain kind of class, then it can only have certain kinds of relationships (defined by the relation properties) with objects. These are also limited by their classes.

Range - For every property p the ontology defines a set of classes $range(p)$. The set $range(p)$ is a set of classes that the property p is limited to have for an object o - defined by the ontology. Given a triple (s, p, o) the ontology requires that o is of the same class of at least one of the classes defined in $range(p)$ for the triple to be consistent.

Domain - For every property p the ontology defines a set of $domain(p)$ classes. The subjects s of every triple (s, p, o) must be of one of the classes defined in $domain(p)$ for the triple to be consistent.

Inconsistency - We characterize an inconsistency in a triplestore as a triple that contains conflicting information with the underlying ontology used to define the classes in which triples must respect. We define two types of inconsistencies organizing the inconsistency classes in two main groups: range violation and domain violation.

- **Range violation Inconsistency** - A triple is deemed inconsistent in its range if, given a triple $t = (s, p, o)$, o for the given p is such that $o \notin range(p)$, such that $range(p)$ the values accepted by p for a relation. Therefore, the triple is range inconsistent when the value of the triple is not part of the range of the predicate.

  Example: Consider the following triple (Lincoln, dbo:birthPlace, University of Alabama). This RDF triple indicates that the birth place of Lincoln is the University of Alabama. However, the property "*dbo:birthPlace*" has a defined range that contains only the ontology class "*dbo:Place*". Therefore, for this property, the object must be of the type "*Place*", which is not the case. The resource "University of Alabama" is of the type "*dbo:University*" and none of it's super classes are subclasses of "*dbo:Place*", turning this triple inconsistent with the ontology.

- **Domain violation Inconsistency** - A triple is deemed inconsistent in its domain if, given (s, p, o), s for the given p is such that $o \notin domain(p)$, such that $domain(p)$ defines the required subjects for the property p according to the underlying ontology. In this sense, the triple is domain inconsistent when the subject s is not part of the possible values that the predicate allows in its ontology.

  Example: Given the triple (Spock, dbo:birthPlace, California). The property "*dbo:birthPlace*" contains as domain only the type "*dbo:Person*". Since the resource "Spock" is of the type "dbo:FictionalCharacter", which is not the type required of the property's domain set, the given triple is considered inconsistent.

3.2 Analysis of inconsistencies in the *DBpedia* update

Based on the defined inconsistency classes, this analysis aimed to measure to which extent the extracted triples in the *DBpedia Live* were inconsistent as inserted in the DBpedia dataset of non-english languages.

The dataset analyzed was the spanish RDF triple dump of ”20190401” that corresponds to april first's dump. We analyzed the first 40000 triples of that dump. We utilized the *DBpedia Extraction Framework*, which is the same utilized in the DBpedia Live extraction, openly available[2]. We also utilized SPARQL queries via *SPARQL-Wrapper* in python to consult the DBpedia full database on information about the resources and properties in the RDF triple dump analyzed.

For detecting range inconsistency, the procedure was as follows: given a triple (s, p, o), we first consult the range definition of the property p. The property in the Spanish dataset only makes reference to its English equivalent, as the Figure 2 shows. In this sense, we consulted their equivalent in the English DBpedia to gather the range sets of the properties. Afterwards, we consult the classes of the object o in the Spanish DBpedia. If none of the classes of o was contained in the set defined in range, then the triple is inconsistent.

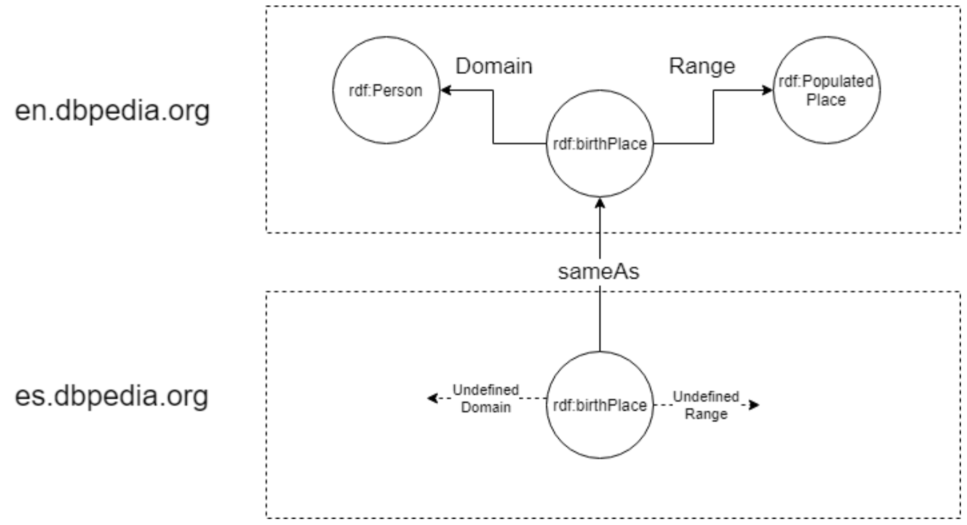

Figure 2: Property definitions of range and domain for the English *DBpedia*.

The same procedure was applied to domain inconsistencies, with the difference that we consult the property's domain instead of range. In this procedure, the type of the subject s is analyzed instead of the object o. This process is presented in the Algorithm 1.

[2]https://github.com/dbpedia/extraction-framework

Algorithm 1: Inconsistency Detection algorithm in DBpedia live extraction.

```
Require: $s, o, p$
1: $range(p) \leftarrow queryPropertyInEnglsihDBpedia(p, range)$
2: if $range(p) \not\subset types(o)$ then
3:     $(s, p, o)$ is range inconsistent
4: end if
5: $domain(p) \leftarrow queryPropertyInEnglsihDBpedia(p, domain)$
6: if $domain(p) \not\subset types(s)$ then
7:     $(s, p, o)$ is domain inconsistent
8: end if
```

The inconsistency detection algorithm was applied after the first extraction in the "mapping based extractor". Therefore, the detection was applied as soon as the triples were formed by the extractor. We then detected for each triple whether it was inconsistent in range, domain or both. The obtained results were organized, separated by each property present in the dataset. We analyzed the frequency in which each property *p* present in the dataset was consistent and the volume of how many triples were inconsistent overall.

Figure 3 presents the results of range inconsistency, whereas Figure 4 shows the results for the domain class inconsistency. Both figures represent each property detected as a dot mapped in the x axis in the amount of inconsistent triples; and in the y axis, the percentage of inconsistent triples for that property.

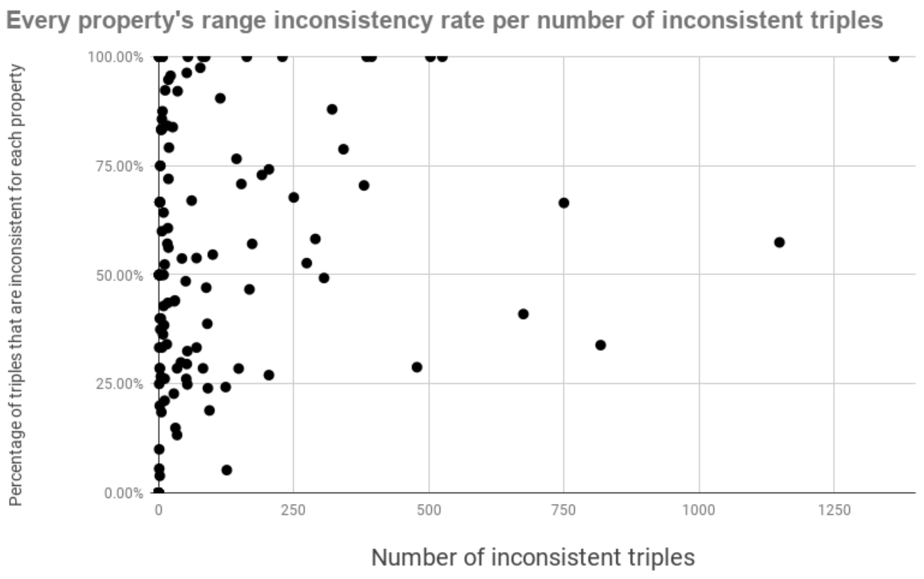

Figure 3: Analysis of property's range inconsistencies

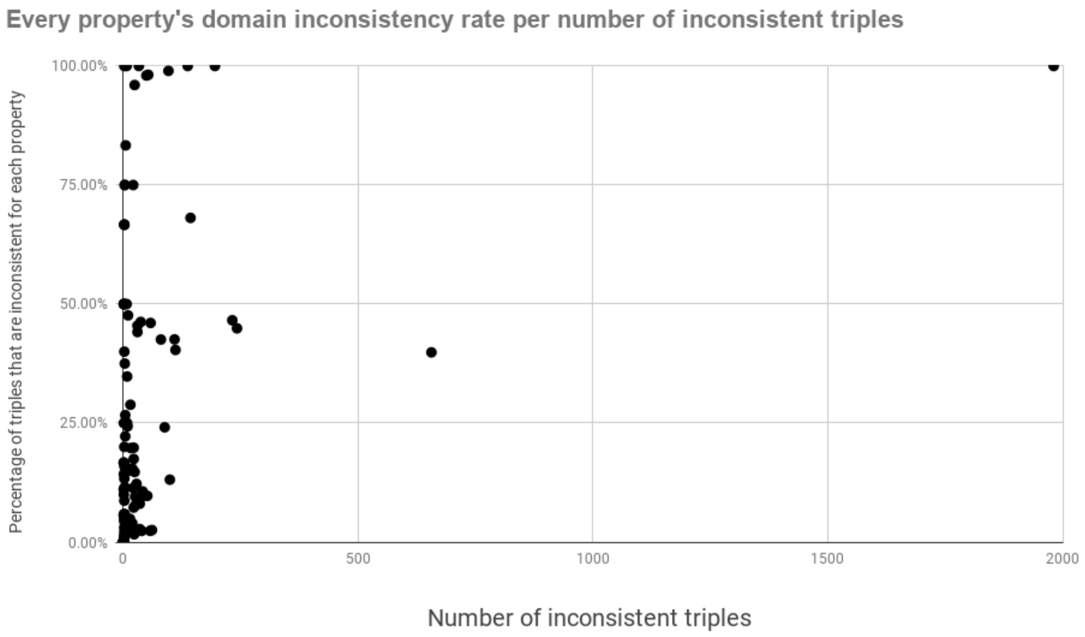

Figure 4: Analysis of property's domain inconsistencies.

Figure 3 and Figure 4 reveal that from all the properties present in the dataset, many of them contain a very high rate of inconsistency. The whole dataset contains 33,58% of triples range inconsistent and 13,38% of domain inconsistent.

In an overall, we observe that most of the detected inconsistencies came from properties with a very high rate of inconsistency. Most of the problems with extractions appear with only a few properties with range and domain definitions not consistent with the property's true lexical meaning or a not good enough mapping of the Wikipedia information into properties.

One glaring example is the property "*dbo:class*" which has a 100% rate of domain inconsistency. The domain set defined for this property is "*dbo:meansOfTransportation*", but the property is repeatedly used to refer to biological classes. The triples and the mappings make complete sense, but the ontological definition doesn't.

Another example of a different mistake causing inconsistency is the property "*dbo:capital*" with 100% range inconsistency. The range set is correctly defined as "*dbo:city*", but none of the object resources were of that type. This highlights another problem with *DBpedia* in secondary languages: not good enough definition of the resources types.

4 Solving the inconsistencies in DBPedia updates

We present our proposed mechanism for inconsistency correction in DBPedia Live (Subsection 4.1) followed by the evaluation conducted to assess the rate of inconsistencies appearing when applying our mechanisms (Subsection 4.2).

4.1 Mechanism for inconsistency correction in DBPedia Live

Our analysis confirmed that the key problems detected in the Spanish DBpedia refer to two aspects: 1) resources which were not properly defined because many of them did not have an specific enough class attribution for the RDF triples to be consistent; and 2) the detection of direct consistency was not possible given that the set and domain definitions for the properties were only present in the English database.

We proposed an algorithm to automate the correction process in the insertion of triple in the extractor's workflow. The algorithm works as a second extraction method operating in the resulting triples from the first extraction. The result is then added with the others as the illustrated in Figure 5.

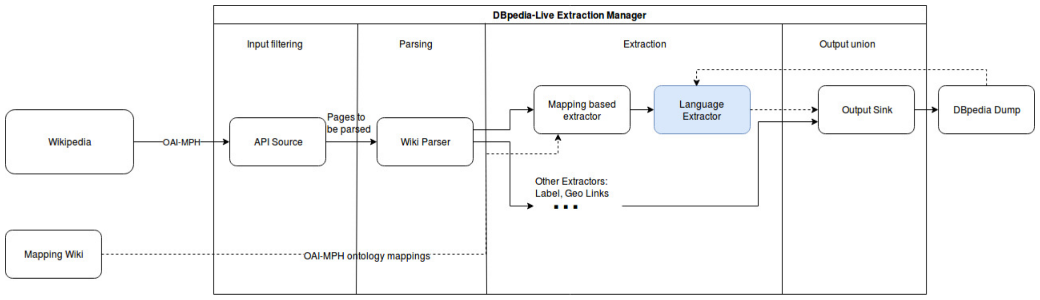

Figure 5: Workflow of the Extraction with our algorithm taking part of the process.

Figure 5 presents that the triples come from the previous extractors and the *query* functions in our algorithm explores information from the DBpedia dump (both the English version and the language under analysis).

The Algorithm 2 presents the implemented procedure for inconsistency detection and correction. The entries of the algorithm are: the triple (containing the subject s, the property p and the object o); α as a threshold of the ratio of inconsistency for a property; β the number of triples that confirms a resource's class/type; γ as a minimum of triples to consider a property relevant for proposing ontological alteration.

Our first approach was to get the resources the most fitting definition from the most updated DBpedia. Therefore, for each resource, we found the equivalent resource on the English DBpedia (if there was one). If the type of the resource was expected (lines 2 and 10 of algorithm 2), then the algorithm transfers that definition to the Spanish DBpedia. In this sense, the algorithm adds the triple "(s, rdf:type, domain)" or “(o, rdf:type, range)” (depending on which resource correction), and then, the triple analyzed would be consistent. If the resource was of some other disjoint type, with the one analyzed, we would consider the triple to be inconsistent.

In the case the algorithm cannot find an equivalent resource, or the equivalent resource did not have any relevant types, it looks into other triples that had referenced the resource in question (lines 4 and 12 in the Algorithm 2). It must find enough triples that assumed the resource was of the type required for the domain/range set. If it does, it confirms that the resource is of that type and adds the triple ”(s, rdf:type, domain/range)”, maintaining its consistency.

The number of triples that confirms a resource of a certain type must be studied because a too small number and we could risk adding inconsistent types to resources because of random human mistakes. If a high number is considered, then we would not resolve enough systematic inconsistencies with the properties. We call the number of triples necessary to confirm a resource of a given type as β. For our application evaluation, the best value empirically found while still maintaining consistency was $\beta = 3$.

At this point, considering we analyzed all possibilities to correct the triple, by properly defining the resources types, the only analysis left was of poorly defined ontology. If a given triple at this point is still considered inconsistent, and the property has a very high rate of inconsistency in an overall, the algorithm raises a flag that the property should be properly looked at and possibly redefined as almost none of the triples containing it are consistent (lines 6 and 14 in the Algorithm 2).

One example of the application of this flag would be to the earlier approached property definition of ”*dbo:class*”, which defined its domain as of the type ”*dbo:meansOfTransportation*”. As every triple containing this property (in the analyzed dataset) would be considered inconsistent, the algorithm must notice this and raise a flag proposing an alteration to the property’s definition in the ontology.

We declared a γ variable in the algorithm to have a standard minimum number of triples before declaring a change in definition of ontology. This variable indicates how many triples are necessary for the algorithm to have previously analyzed before deciding it can declare an ontology change. In our experiments, the obtained value of γ was 15.

As for the definition of a ”very high rate”, it has to be made empirically. On our first analysis, we could observe that the considered poorly defined properties had a rate of 100% inconsistency either in range or domain. Considering the corrections provided and that there is a natural number of inconsistencies in a very large sample size of triples, we assumed safe to consider the “very high rate” as being higher than 90%. In the algorithm, we refer to it as the cutting rate of ontology flag and named it α. Therefore, if a triple is considered inconsistent of some kind, and in that kind, the property has an inconsistency rate higher than α, then the flag is lifted for a human to check.

Algorithm 2: Language Extractor algorithm for non-english extraction.

```
Require: $s, o, p, \alpha, \beta, \gamma$
    {Range Inconsistency detection}
 1: if $range(p) \not\subset types(o)$ then
 2:    if $\exists(o, rdfs : sameAs, range(p))$ then
 3:       $return \leftarrow$ add triple $(o, rdf : type, range(p))$
 4:    else if $N > \beta|$ N is number of $(x, y, o)|range(y) = range(p)$ then
 5:       $return \leftarrow$ add triple $(o, rdf : type, range(p))$
 6:    else if inconsistency rate$(p) > \alpha$ & number of triples with $p > \gamma$ then
 7:       raise flag for ontology update for $p$
 8:    end if
 9: end if
    {Domain Inconsistency detection}
10: if $domain(p) \not\subset types(s)$ then
11:    if $\exists(s, rdfs : sameAs, domain(p))$ then
12:       $return \leftarrow$ add triple $(s, rdf : type, domain(p))$
13:    else if $N > \beta|$ N is number of $(s, y, x)|domain(y) = domain(p)$ then
14:       $return \leftarrow$ add triple $(s, rdf : type, domain(p))$
15:    else if inconsistency rate$(p) > \alpha$ & number of triples with $p > \gamma$ then
16:       raise flag for ontology update for $p$
17:    end if
18: end if
```

4.2 Evaluation of inconsistency correction in DBPedia

Our second analysis applies the same consistency check previously defined to the results of the same dataset after it has been treated by our algorithm. We explored the same datasets as in the first analysis. The objective is to understand how inconsistent the resulting dataset remains and compare it to the one generated without the solution algorithm.

Table 1 presents the results of our evaluation.

Our second analysis shows that the final triple set represented considerable improvement. Each of the types of inconsistencies detected before has considerably decreased. Range inconsistency, from 33,58% to 17,35%, showed a 50,62% reduction of inconsistencies. Domain inconsistencies dropped from 13,38% to 7,62%, with a reduction of 56,85%.

We tackled the most inconsistent properties with each of the two steps of our algorithm. If we look at the inconsistency rate for each property, we observe a decrease of high rate of inconsistency properties in Figure 6 and Figure 7. Although a percentage of inconsistencies remain, the solution showed to be useful to make *DBpedia* significantly more consistent.

Original DBpedia Live Extraction		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	13434	33.58%
Domain Inconsistency	5355	13.38%

Extraction With our proposed Algorithm		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	6941	17.35%
Domain Inconsistency	3046	7.62%

Table 1: Results of the datasets with 40000 triples.

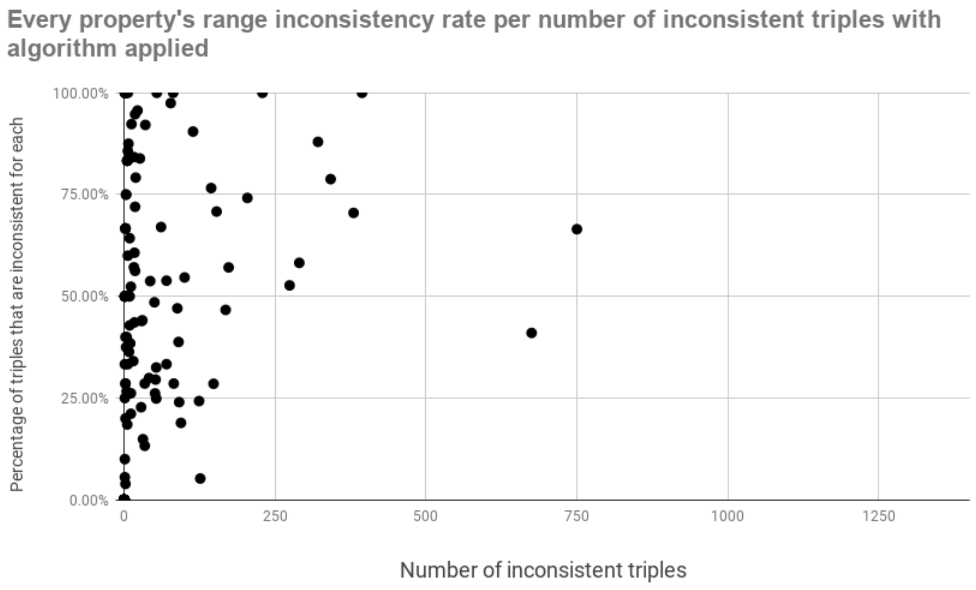

Figure 6: Property's range inconsistency rate and number of inconsistent triples after applying the algorithm.

The properties that mostly contributed to the previous dataset's inconsistencies were handled. The percentage of the most of the properties dropped. Even though some still have a high percentage rate of inconsistency per property, they are less expressive in volume.

.

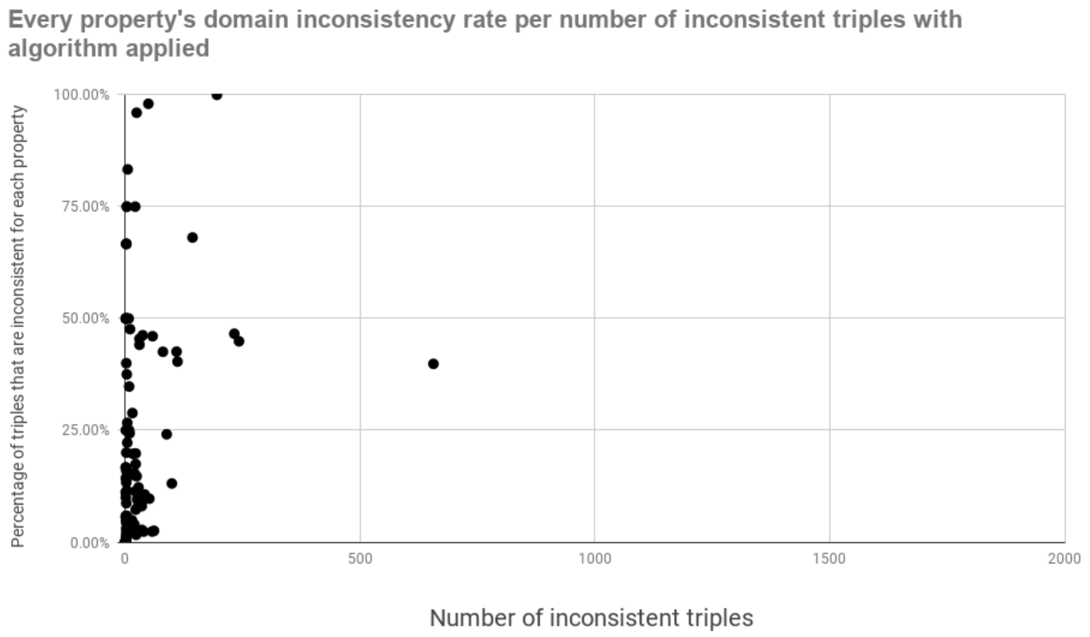

Figure 7: Property's domain inconsistency rate and number of inconsistent triples after applying the algorithm.

5 Discussion

This work showed results of improving consistency in the evolution of multi-chapters of the DBpedia. The proposal is to help making all DBPedia language chapters more trustworthy.

The two types of addressed inconsistencies are the two most present in the DBpedia. In an overall, there are other inconsistency definitions such as class disjointedness, language, etc. However, our exploratory analysis showed that they represented a very small percentage of all inconsistencies.

Some inconsistencies still remain open given the difficulties involved in their detection. Most of the solution relies on researching a resource in the English DBpedia. When the foreign language version of that resource does not have a mention to it's English equivalent, the solution cannot track the properties, limiting the algorithm's effectiveness

Based on the results obtained in our experiments, we indicate that DBpedia in other languages different than English is still very inconsistent. The idea of inconsistency is not used effectively and allows any triple to be inserted. Our findings signalized that for the DBpedia to be usable in every language, it requires further attention to the resource types and tools to turn DBpedia overall more inclusive in it's consistency.

We offered a solution to keep other language DBpedia datasets as consistent as the English one, by applying their same methods of inconsistency checking. The proposed method was effective by reducing on average 50% of all the inconsistencies of a given dataset. Most importantly, the solution requires minimum human work and can be easily scalable.

The human work needed is periodic and simple. The algorithm indicates properties requiring attention to update their definition. Naturally, as more triples go through the extractor, the work is reduced to a minimum, where the tool becomes nearly fully automated.

In addition, the solution investigated can be applied to a larger number of RDF datasets in addition to the DBpedia, such as Wikidata. As the tool identifies and proposes a solution for any RDF dataset that has a defined ontology and a well established amount of data present. It only needs to be applied to an extractor. There are no specific limitations to the tool that are specific to DBpedia.

The tool also, if utilized in the English version of DBpedia could improve their consistency ratio. Although the core of the work was based on the difference between the secondary languages and the main DBpedia, our algorithm of detecting resources' types through other triples could be used on the English version.

In future work, we plan to explore the correction of other types of inconsistencies, such as, inconsistencies caused by update of previous data and resources unlinked to their other language counterparts - that can be applied to the extractor on every language (including english).

6 Conclusion

This work showed that DBpedia in languages other than English is not treated with the same attention, leading to an inconsistent RDF dataset. It is important that DBpedia remains consistent over time so it can be effectively used. Its use in other languages can only be enforced if such dataset is consistent. This work offered a solution to decrease the number of inconsistencies. Our evaluation showed the effectiveness in greatly reducing the inconsistencies. Future work aims to address other types of inconsistencies using our solution as a basis.

Acknowledgements

This work was financially supported by the São Paulo Research Foundation (FAPESP) (grants #2017/02325-5 and #2018/06444-1)[3].

[3]The opinions expressed in here are not necessarily shared by the financial support agency.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[2] Tim Berners-Lee Christian Bizer, Tom Heath. Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, 5:1–22, 2009.

[3] Fabien Gandon Elena Cabrio, Serena Villata. Classifying inconsistencies in dbpedia language specific chapters. *9th International Conference on Language Resources and Evaluation*, pages 1443–1450, 2014.

[4] Grigoris Antoniou-Vassilis Christophides George Konstantinidis, Giorgos Flouris. A formal approach for rdf/s ontology evolution. *18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 70–74, 2008.

[5] Harald Sack Gerald Topper, Magnus Knuth. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems (I-SEMANTICS' 12)*, pages 33–40. I-SEMANTICS, 2012.

[6] Aldo Gangemi Heiko Paulheim. Serving dbpedia with dolce - more than just adding a cherry on top. *14th International Semantic Web Conference (ISWC 2015)*, pages 180–196, 2015.

[7] Fabrizio Orlandi Sören Auer Simon Scerri Kemele M. Endris, Sidra Faisal. Interest-based rdf update propagation. *14th International Semantic Web Conference (ISWC 2015)*, pages 513–529, 2015.

[8] Jens Lehmann Soren Auer Sebastian Hellmann, Claus Stadler. Dbpedia live extraction. *Meersman R., Dillon T., Herrero P. (eds) On the Move to Meaningful Internet Systems: OTM 2009. Lecture Notes in Computer Science*, 5871:1209–1233, 2009.

[9] Saeedeh Shekarpour Sören Auer Maria-Esther Vidal Sidra Faisal, Kemele M. Endris. Co-evolution of rdf datasets. *Lecture Notes in Computer Science*, 9671.

[10] Heinrich Herre Soren Auer. A versioning and evolution framework for rdf knowledge bases. *Lecture Notes in Computer Science*, pages 55–69, 2006.

[11] Gildas Ménier Pierre-François Marteau Youen Perón, Frederic Raimbault. On the detection of inconsistencies in rdf data sets and their correction at ontological level. pages 1–11, 2011.