Handling Multi-chapter Inconsistencies in DBpedia Evolution

Julio Cesar dos Reis[1] **, Tulio Brandão Soares Martins**[1]

[1]Institute of Computing, University of Campinas (Unicamp), Campinas, Brazil

jreis@ic.unicamp.br, tuliomartinstm1523@gmail.com

Resumo. *DBpedia é um enorme recurso disponível na Web of Data. É relevante atualizar este conjunto de dados com base em novas informações que aparecem na Wikipedia. No entanto, essa operação pode fornecer inconsistências no conjunto de dados para capítulos em diferentes idiomas. Embora a literatura existente tenha definido ferramentas para atualização do DBpedia, faltam estudos relacionados à compreensão e detecção de inconsistências multicapítulos em sua evolução. Neste artigo, definimos um conjunto de classes de inconsistência relacionadas a mudanças triplas no DBpedia para informar uma ferramenta de software adequada para detectar instâncias dessas classes quando novos dados são atualizados para todas as linguagens, removidos ou inseridos no conjunto de dados. Demonstramos tipos de inconsistências que aparecem na evolução do DBpedia e propomos uma solução para corrigir essas inconsistências. Nossa avaliação avaliando a técnica proposta revela sua utilidade. Nossos resultados mostram que as classes de inconsistências propostas combinadas com a solução definida podem tornar o DBpedia mais confiável.*

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 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 understanding and detecting multi-chapter inconsistencies in its evolution. In this paper, we define a set of inconsistency classes related to triple changes in DBpedia to inform a software tool suited to detect instances of these classes when new data is updated for all languages, removed or inserted in the dataset. We demonstrate types of inconsistencies appearing in the evolution of DBpedia and propose a solution to correct these inconsistencies. Our evaluation assessing the proposed technique reveals its usefulness. Our results show that the proposed classes of inconsistencies combined with the defined solution can make DBpedia more reliable.*

1. Introduction

The continuous growth of the Web raises several questions on how to handle big chunks of related data. 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 [Auer et al. 2007] was proposed to take information from Wikipedia and make it machine readable.

DBpedia has become a relevant tool that collects 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 over time. 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* [Sebastian Hellmann 2009], 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 all languages. For instance, some triples that exist in a dataset are not consistent with the ontology in place. In addition, many resources in languages other than English 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 undesirable results.

The research on how to make *DBpedia* a reliable knowledge base is extensive and continuous. The current literature has approached inconsistency classification [Gerald Topper 2012] [Elena Cabrio 2014] [Youen Perón 2011] and data co-evolution [Faisal et al. 2016] [Kemele M. Endris 2015] [George Konstantinidis 2008]. Some investigations have approached language specific inconsistencies [Elena Cabrio 2014]. However, to the best of our knowledge, it lacks studies applied to the live extraction to filter inconsistencies in real time updates for distinct *DBpedia* language datasets.

In this article, we formally define a set of inconsistency classes that can occur during the *DBpedia Live* extraction. 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 related work. Section 3 defines the types of inconsistencies and proposes a solution to correct the inconsistencies and avoid their recurrence. Section 4 presents the evaluation results. Section 5 discusses our obtained findings. Finally, Section 6 presents the final considerations.

2. Related Work

Pern *et al.* [Youen Perón 2011] defined inconsistencies in RDF data and proposed 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* [Sebastian Hellmann 2009].

Cabrio *et al.* [Elena Cabrio 2014] 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 languages with multiple references of one resource to others.

Topper *et al.* [Gerald Topper 2012] 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 such an idea in the proposition of updating specific definitions of the ontology, but most of the correction techniques proposed in our work require small alterations 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 [Heiko Paulheim 2015] 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.

Literature has approached the problem of updating RDF graphs, addressing mostly the precautions that should be taken, how they must obey ontology axioms and possible inconsistencies that can be caused through these updates. Endris *et al.* [Kemele M. Endris 2015] explored how the local duplication of data can eventually cause inconsistencies in RDF graphs such as *DBpedia* during its updates. In addition, Konstandinidis *et al.* [George Konstantinidis 2008] 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*.

Auer & Herre *et al.* [Auer and Herre 2007] 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 in the triple addition operation, by specifically applying it to the DBpedia Extraction Manager for the *DBpedia Live*.

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

The *DBpedia Live platform* [Sebastian Hellmann 2009] aims to extract triples without requiring one single dump. Our work focused on the mapping extractor, which is the extractor utilized when the added information of resources 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. 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* over time and helps make this RDF dataset further reliable.

3. Solving inconsistencies in DBpedia updates

We present formal definitions followed by our proposed mechanism for inconsistency correction in *DBpedia Live*.

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 types 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 subject 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 respect to the underlying ontology used to define the classes. 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 its 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 $o \notin domain(p)$. The $domain(p)$ 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.

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

We proposed an algorithm to automate the correction process during the insertion of triples in the extractor's workflow. The algorithm works as a second extraction method operating over the resulting triples from the first extraction. The result is then added with others as illustrated in Figure 1[1].

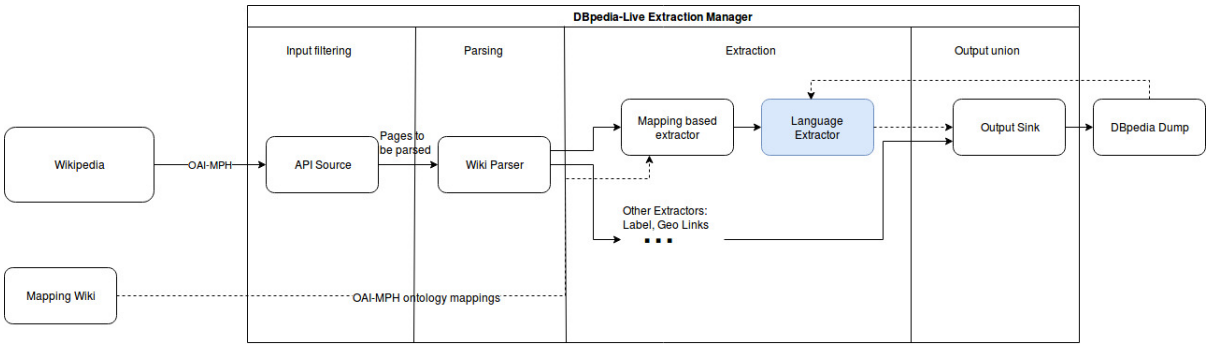

Figure 1. Workflow of the Extraction with our algorithm taking part of the process.

Figure 1 presents that triples come from the previous extractors and the $query$ functions in our algorithm explores information from the DBpedia dump (for all chapters of *DBpedia*). Algorithm 1 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 ontology alteration.

Our first approach was to retrieve resources with the most fitting definition from the most updated *DBpedia* release. Therefore, in the case of a non-english chapter extraction, 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 1), then the algorithm transfers that definition to the *DBpedia* chapter under analysis. In this sense,

[1] `https://gitlab.ic.unicamp.br/jreis/dbpedia-consistency`

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 that 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 Algorithm 1), applying such method for every chapter. 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. 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 resource 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 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 Algorithm 1).

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 analyses, we observe that those 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 raised for a human to check.

4. Evaluation

This assessment applies our proposal (Algorithm 1) in three datasets. Based on the obtained results, we applied our consistency check procedure. The objective is to understand how inconsistent the resulting datasets remain and compare them to the one generated without our solution Algorithm 1. Table 1 presents the results of our evaluation.

Algorithm 1 Modified Extraction algorithm for all DBpedia chapters.

Require: $s, o, p, \alpha, \beta, \gamma$

{Range Inconsistency detection}

if $range(p) \not\subset types(o)$ **then**
  if $\exists(o, rdfs : sameAs, range(p))$ **then**
    $return \leftarrow$ add triple $(o, rdf : type, range(p))$
  else if $N > \beta|$ N is number of $(x, y, o)|range(y) = range(p)$ **then**
    $return \leftarrow$ add triple $(o, rdf : type, range(p))$
  else if inconsistency rate$(p) > \alpha$ & number of triples with $p > \gamma$ **then**
    raise flag for ontology update for p
  end if
end if

{Domain Inconsistency detection}

if $domain(p) \not\subset types(s)$ **then**
  if $\exists(s, rdfs : sameAs, domain(p))$ **then**
    $return \leftarrow$ add triple $(s, rdf : type, domain(p))$
  else if $N > \beta|$ N is number of $(s, y, x)|domain(y) = domain(p)$ **then**
    $return \leftarrow$ add triple $(s, rdf : type, domain(p))$
  else if inconsistency rate$(p) > \alpha$ & number of triples with $p > \gamma$ **then**
    raise flag for ontology update for p
  end if
end if=0

Based on the defined inconsistency classes, our analysis aimed to measure to which extent the extracted triples in the *DBpedia Live* remained inconsistent as inserted in the *DBpedia* dataset of all languages, emphasizing non-english ones, the most neglected chapters of *DBpedia*.

The datasets analyzed were three: the Spanish, French and English RDF triple dumps of "20190401" that corresponds to April first's dump. We analyzed the first 40000 triples of the Spanish dump, 20272 triples of the french chapter and 22400 triples of the English chapter. 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 *SPARQLWrapper* in Python to consult the DBpedia full database for 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 non-english datasets only makes reference to its English equivalent. In this sense, we consulted their equivalent in the English *DBpedia* for the extraction of non-english. A similar procedure was applied for domain inconsistency check.

Our evaluation shows that the final triple set represented considerable improvement in all chapters analyzed (English, Spanish and French). Each of the types of inconsistencies detected before has considerably decreased. In the Spanish chapter, range inconsistency decreased from 33.58% to 17.35%, by showing a 50.62% reduction of in-

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

Table 1. Results of all three datasets.

Original DBpedia Live Extraction		
Spanish Chapter		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	13434	33.58%
Domain Inconsistency	5355	13.38%
French Chapter		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	5804	28.63%
Domain Inconsistency	2638	13.01%
English Chapter		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	4568	20.39%
Domain Inconsistency	2073	9.25%
Extraction with our proposed Algorithm		
Spanish Chapter		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	6941	17.35%
Domain Inconsistency	3046	7.62%
French Chapter		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	2993	14.76%
Domain Inconsistency	1096	5.41%
English Chapter		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Range Inconsistency	2835	12.66%
Domain Inconsistency	1248	5.57%

consistencies. Domain inconsistencies dropped from 13.38% to 7,62%, with a reduction of 56.85%. In the French chapter, the reduction was also very close to 50%.

We can observe, however, that our solution is more effective when operating with chapters other than the English one. As the resources are more defined and there is less chance of choosing a property incorrectly, the correction itself is also less efficient in the English chapter. Still, it managed to reduce the overall consistencies close to 39% overall, which is an improvement compared to the current extraction mechanism.

.

5. Discussion

This investigation showed results of improving consistency in the evolution of multilingual chapters of the *DBpedia*. The proposal aimed to make *DBPedia* language chapters more trustworthy. The two types of addressed inconsistencies are the two most present in the *DBpedia* updates.

Based on the results obtained in our experiments, we indicate that the update of *DBpedia* (via the Live platform) contributes in the generation of inconsistencies. This

is even more for less defined and robust chapters. The consistency verification is not effectively explored in these chapters and several inconsistent triples are inserted. Our findings indicated that further attention and studies are required on novel mechanisms and tools to keep the *DBpedia* chapters up-to-date in a consistent way. This is of key relevance to enable the use of *DBpedia* in multiple languages.

We offered a solution to keep multi-chapters of *DBpedia* datasets as consistent as the English one as well as improve the English extraction. The use of our method enables the detection and correction of inconsistencies based on the definition of domain and range of predicates. 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 utilization of our proposal in the English version of *DBpedia* further improved their consistency ratio. Although the core of the work was based on the difference between the secondary languages and the main DBpedia, our algorithm for detecting resources' types through other triples worked well in the English version, reducing overall inconsistency by 40%.

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

6. Conclusion

Web of data with very large RDF datasets require adequate treatment for their growing with consistency over time. It is important that DBpedia remains consistent over time so it can be effectively used. Its use in all languages can only be enforced if the chapters keep consistent with one other. This work proposed a solution to verify and correct inconsistencies related to the live extraction mechanism for the DBpedia update inserts inconsistent triples, specially in languages other than English. Our evaluation showed the effectiveness in greatly reducing the inconsistencies generated. Future work aims to address other types of inconsistencies using our solution as a basis.

Acknowledgments

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

References

[Auer et al. 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Auer and Herre 2007] Auer, S. and Herre, H. (2007). A versioning and evolution framework for rdf knowledge bases. *Perspectives of Systems Informatics*, pages 55–69.

[3]The opinions expressed in this work do not necessarily reflect those of the funding agencies.

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

[Faisal et al. 2016] Faisal, S., Endris, K. M., Shekarpour, S., Auer, S., and Vidal, M.-E. (2016). Co-evolution of rdf datasets. *Web Engineering*, pages 225–243.

[George Konstantinidis 2008] George Konstantinidis, Giorgos Flouris, G. A. V. C. (2008). A formal approach for rdf/s ontology evolution. *18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 70–74.

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

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

[Kemele M. Endris 2015] Kemele M. Endris, Sidra Faisal, F. O. S. A. S. S. (2015). Interest-based rdf update propagation. *14th International Semantic Web Conference (ISWC)*, pages 513–529.

[Sebastian Hellmann 2009] Sebastian Hellmann, Claus Stadler, J. L. S. A. (2009). 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.

[Youen Perón 2011] Youen Perón, Frederic Raimbault, G. M. P.-F. M. (2011). On the detection of inconsistencies in rdf data sets and their correction at ontological level. pages 1–11.