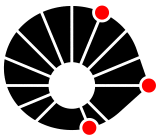

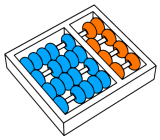

Addressing Inconsistencies in the DBPedia Evolution

Túlio Brandão Soares Martins, Julio Cesar dos Reis

Relatório Técnico - IC-PFG-19-36
Projeto Final de Graduação
2019 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Addressing Inconsistencies in the DBPedia Evolution

Túlio Brandão Soares Martins, Julio Cesar dos Reis*

December 2019

Abstract

DBpedia is a huge RDF dataset built upon the extraction of information from Wikipedia pages. It provides data converted to RDF linked to several other datasets by fully implementing the concepts of Linked Data. *DBpedia* needs to be constantly updated overtime to keep its usability. However, this process is not without its flaws. Much of the extraction performed generates inconsistencies in the RDF triples. Despite many studies in the field of ontology and linked data, there is a lack of studies to improve the *DBpedia Live Extraction*, specifically as an updating tool for *DBpedia*. In this work, we propose the definition of certain types of inconsistencies that are not usually approached in literature. We offer methods for their detection and correction implemented in the core of the *DBpedia Live* extraction process. Our study shows the relevance of our proposal as a progress obtained for both the definition of RDF triple inconsistencies as well as an improved software tool to turn *DBpedia* further reliable.

1 Introduction

With the growth of information on the internet, a movement to connect all the information of the web through relational data. This movement, called Linked Data, meant to annotate most of the information available on the web and create relationships between them by creating the so called Web of Data. One particular idea to apply this was the creation of *DBpedia*. The *DBpedia* project [1] was proposed to take information from Wikipedia and turn it machine readable and completely linked.

Considering that it is based on the largest free knowledge encyclopedia on the web, the *DBpedia* gained importance on the world of web of data and had its utilities. There were initial projects utilizing the database such as *DBpedia Mobile*, a

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

cellphone application to observe locations on a map using the data from *DBpedia* to find relations between important places. There is also the use of making very complex queries in a database containing all of the information of Wikipedia.

As Wikipedia grows over time, so the same must occur in the DBpedia. One important tool that helps keeping the database updated with current information is the *DBpedia Live* [9] - a tool that queues up all of the new information that is being added to Wikipedia and converts it into the annotated format of data used in DBpedia - the RDF.

Considering the size and relevance of *DBpedia*, the conversion of Wikipedia information into RDF must be as reliable as possible. Yet, many of the RDF triples that are added are considered inconsistent with the underlying ontology used. Some of the inconsistencies generated in the extraction process include the addition of conflicting information during updates and duplication of resources.

Given the importance of a clean and reliable extraction, our work focuses on detecting inconsistencies caused in the DBpedia Live extraction tool. Our goal is to clearly define the inconsistencies that occur in the process of converting information into RDF. We offer a solution implemented in the *DBpedia Live Extraction Framework*, the tool utilized by DBpedia Live to convert the data.

In our methodology, we first establish the definition of the explored classes of inconsistencies. Then, we implement an algorithm to detect those inconsistencies to analyze the root cause of these inconsistencies. We measure how frequently they are present in the extraction process. Finally, we propose an algorithm to correct the inconsistencies generated by the tool automatically. The solution works without requiring human supervision to guarantee the efficiency of the Live extraction.

Our work demonstrates an improvement to the current extraction process, providing a new detection method implemented in the extraction tool as well as an algorithm of correction. In addition, our definitions of inconsistencies can push the boundaries of the current literature on RDF inconsistencies even further. Most importantly, as the goal of our research, the addition of our algorithms to the extraction prove to turn DBpedia more reliable.

The remaining of this document is organized in the following structure: Section 2 presents fundamental concepts related to the study of Linked Data and RDF triples as well as related work on the field. Section 3 presents the proposed methods to detect and correct the inconsistent triples. Section 4 describes the experiments we executed and the results obtained, while 5 discusses the results and implications of the experiments and algorithms elaborated. Finally, section 6 argues our final thoughts on the study.

2 Background

DBpedia Live [9] is a tool that performs the extraction of new Wikipedia infomration and converts it into RDF triples based on pre-existing mapping. The Live framework 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 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.

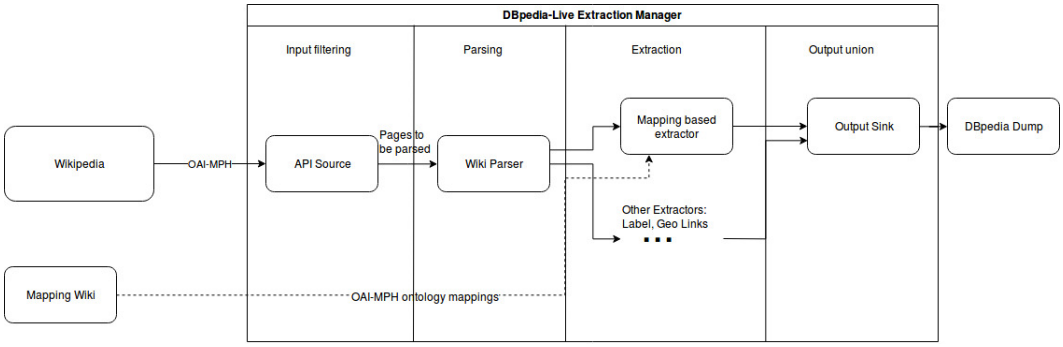

Figure 1: Workflow of the *DBpedia Live Extraction*.

Some of the related work to our research can be found in Bizer & Heath [3], which first addresses Linked Data, its objectives, and goes into further details on the overall movement started by the W3 Schools. This paper goes into detail about the

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

relevance of Linked Data as well as mentioning *DBpedia* as a key participator, giving good context to our research, however, doesn't enter the subject of inconsistencies.

Pern *et al.* [11] defined RDF inconsistencies in general. The idea was a basis to our research in terms of defining what an inconsistency is in the context of RDF databases, one that was more developed by Cabrio *et al.* [4] proposed the classification of inconsistencies regarding different language databases. The latter research closely approaches to our proposal in this paper as it defined the language inconsistency and mapped ways of correcting such inconsistencies we handled in our studies.

Topper *et al.* [5] worked in the detection of inconsistencies in DBpedia. The authors proposed the correction of certain inconsistencies by improving the current DBpedia ontology as well as correctly defining and detecting them. Their methods of detection work similarly as ours, however, their proposition of altering the ontology to improve DBpedia differs from our approach of altering simply the extraction tool.

Paulheim & Gangemi [6] proposed an ontology enhancement to improve DBpedia's consistency. The authors proposed the addition of DOLCE to the ontology to find systematic errors that are present in the current database. Our study once again proposes no alterations of the ontology format to handle the inconsistencies defined in this paper, focusing only on improving the extraction to make less mistakes autonomously.

Auer & Herre *et al.* [2] defined the alteration of ontology together with the evolution of the RDF database to make any RDF database that is live updated more consistent. Utilizing change operations and generating metainformation regarding changes in the ontology, their work aimed to better treat the consistency of RDF databases. Though unlike our approach to solve consistency in DBpedia, their studies on how RDF data evolves through time was very relevant to our paper, specially when studying inconsistencies caused by the natural evolution of DBpedia.

Regarding the idea of inconsistencies being generated with the live update of an RDF database, 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. This idea was taken into consideration when analysing the inconsistencies caused by the natural evolution of the database where one or more new triples being added can cause an inconsistency of conflicting or contradictory data. We have specifically defined one type of inconsistency based on that idea and discuss it in more detail later on the paper.

The same dilemma of RDF database evolution is approached again by Faisal & Endris *et al.* [10] only this time taking into account the co-evolution of data, where more than one instance of the database is being updated simultaneously and the research proposes a way to avoid causing inconsistencies in such cases. Though not directly handling the same cause of inconsistencies as our paper, our research was inspired by their approach when developing a solution to handle the triples that broke consistency of the database.

Martins and Dos Reis [8] studied semantic inconsistencies, how to detect and correct them. The study sought to improve the extraction tool for the *DBpedia Live* much like this one - by defining types of inconsistency and developing a method to detect and correct them *on-the-fly*. The difference lies in the types of inconsistencies defined and handled. Whereas in our previous work we tackled semantic inconsistencies and their presences in different chapters of DBpedia, this time we tackle different types of inconsistencies - evolution and language inconsistencies.

As a refinement of our previous work [8], we propose to handle inconsistency detection and classification in the context of the *DBpedia Live extraction manager*. In this study, we advance the research a step forward defining new types of inconsistencies and propose new methods for their detection and correction. We believe that this added information helps making DBpedia a more reliable RDF dataset by pushing forward the research of RDF inconsistencies. This could implicate in the DBpedia gaining more relevance and utility.

3 DBPedia's Evolution and Language Inconsistencies

We start by addressing the format of data annotated in DBpedia, the RDF - Resource Description Framework, composed by triples. A triple is defined as follows:

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 have 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.

Inconsistent Triples - We define an inconsistency in the RDF database if there are one or more triples that do not obey the ontology in place. A triple is deemed inconsistent to the dataset if, given its current state, the triple is inconsistent to the ontology given the s and o already defined. Therefore, the addition of the triple would cause the RDF database to be inconsistent.

With these definitions of inconsistency, this investigation approaches two specific types of inconsistencies: **Evolution Inconsistency** and **Language Inconsistency**.

We propose algorithms to detect and correct inconsistent triples based on the defined classes. Our solution works on the level of the *DBpedia Extraction Framework*

which is the same utilized in the *DBpedia Live* extraction, openly available[2]. All of our algorithms were inserted in the *Extraction Manager*. Figure 2 illustrates where in the *Extraction Manager* our algorithms are inserted.

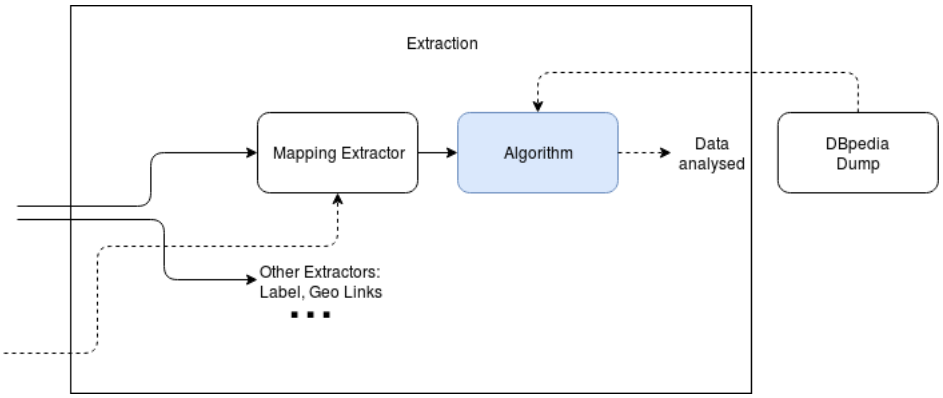

Figure 2: Detection algorithm in the context of the *DBpedia Live Extraction* Workflow.

The data analysed is the result of the Mapping Extractor, the extractor responsible for handling triples composed by s p and o (subject, predicate and object), such that o is necessarily another resource in the DBpedia.

To detect both occurrences of inconsistencies we separate both algorithms of detection, one for each type defined before.

3.1 Evolution Inconsistency

A triple is considered evolutional inconsistent if the triple itself is semantically correct, and the classes of s and o are consistent with the ontology. However, due to other triples present in the database, the addition of the triple would cause an inconsistency in the database.

For example, if a triple being added declares (Bill Gates, dbp:residence, Medina_,Washington), and another triple is to be added declaring a different residence in a non-coexistent location such as (Bill Gates, dbp:residence, dbo:Los_Angeles) there is an occurrence of information that has been updated and the simple addition of the triple would cause inconsistency (with the correct procedure being the replacement of the previous triple). Another similar of such inconsistency would be if the resource happen to be declared deceased by other triples (such as with a property dbp:deathPlace or dbp:deathDate) which would be inconsistent to a resource still having a triple of the type dbp:residence. This example is better illustrated in Figure 3

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

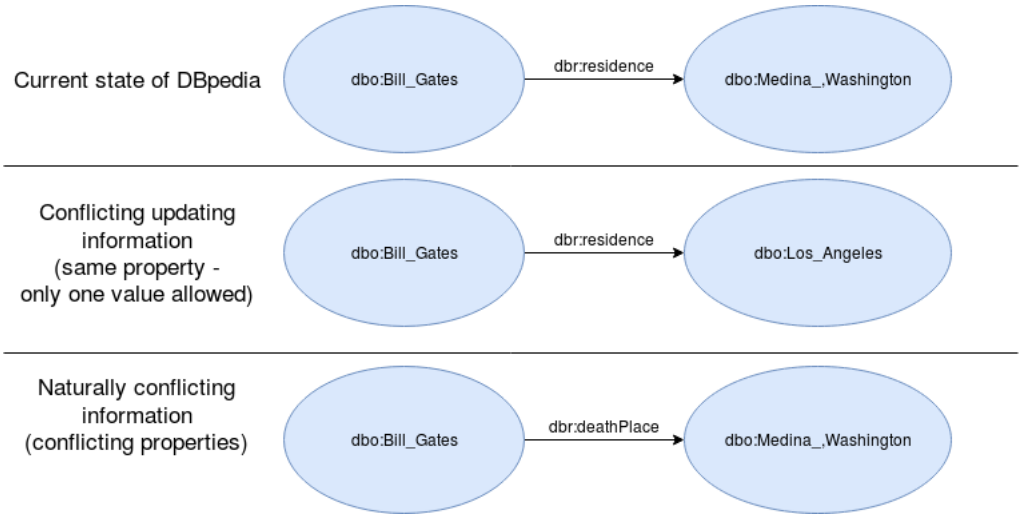

Figure 3: Example of conflicting triples being added and the current state of *DBpedia*.

For the detection of the Evolution Inconsistency, it is necessary to make queries in the *DBpedia* Dump regarding both resources involved on the new triple added - the resources being the subject (s) and the object (o) and analyze each one individually.

First, we analyzed which properties being inserted already contained a triple of the same property being inserted and verified not only with the ontology but with overall triples of the same property, if it was present more than once for a determined resource. This way the algorithm would avoid adding more than one triple in a scenario where the replacement of a previous one should be made. One example of this are triples that indicate the current location of a resource, in the case of multiple triples indicating more specific places, it is valid, however if they declare places that don't naturally share a common location then there is an inconsistency and a replacement of the triple should have taken place.

Secondly, for every triple added, it was verified for triples of the type "rdf:type" to check if the addition of the triple could eventually lead to class disjointedness. The verification was made by crossing references of the current triples to one given resource and checking if the type being added is consistent to the resource triples already present.

Finally, we verified if a triple being added could be considered inconsistent, not by ontology definition, but by verification of triples of the same property. We discovered the frequency in which triples appeared for every type and crossed with the type of the resource to verify if the resource in question (both s and o) can coexist with the triples it currently has. One simple example of this was, given the addition of a triple of property p "dbp:parents" the algorithm would then search for similar triples of the property and discover that if there were any triples of type "dbp:child" then they

would have to be mirror triples of each other, the s of one is the o of another and vice-versa.

Algorithm 1 presents this procedure in details.

Algorithm 1: Evolution inconsistency detection algorithm.

Require: s, p, o
1: $frequency(p) \leftarrow queryFrequencyOfPForOneResource(p)$
2: **if** $frequency(p) = 1$ **and** p is duplicate for s **or** o **then**
3: s, p, o is evolutional inconsistent
4: **end if**
5: **if** $p =$ "$rdf : type$" **then**
6: $possibleTypesOfResource(s) \leftarrow queryAllTriplesFromResource(s)$
7: **if** $o \not\subset possibleTypesOfResource(s)$ **then**
8: s, p, o is evolutional inconsistent
9: **end if**
10: **end if**
11: $correlation(s, p) \leftarrow querySimilarTriplesFromResource(s, p)$
12: $correlation(o, p) \leftarrow querySimilarTriplesFromResource(o, p)$
13: **if** $correlations(s, p)$ is inconsistent **or** $correlations(o, p)$ is inconsistent **then**
14: s, p, o is evolutional inconsistent
15: **end if**

Applying the algorithm for detection we then proceed to the correction phase. Given we had a method of identification of inconsistencies, the algorithms for corrections utilized the same procedure to know which triples needed to be handled. Based on that, the algorithm made the decision to add another triple to make the information consistent.

The Evolution Inconsistency correction occurs simply by replacing the declaration of inconsistency with the correct method of correction when possible. For the first case of a single triple being necessary, the replacement of the old triple is simply applied. For the definition of a class that will cause inconsistency there is no other type that can guarantee the retention of meaning and still be consistent with DBpedia in place, so no automatic solution was offered.

For the inconsistency detected by correlation of other triples, the cases vary as for each property a correlation is detected. For when possible, a triple was either added to maintain consistency with other resources or replaced by a new one detected by correlation

Algorithm 2: Evolution inconsistency correction algorithm.

Require: s, p, o

1: $frequency(p) \leftarrow queryFrequencyOfPForOneResource(p)$
2: **if** $frequency(p) = 1$**and**p is duplicate for s**or**o **then**
3: $return \leftarrow$ remove triple detected as duplicate
4: **end if**The secondary part of detection no proper solution was devised so the correction algorithm only deals with the other two methods
5: $correlation(s, p) \leftarrow querySimilarTriplesFromResource(s, p)$
6: $correlation(o, p) \leftarrow querySimilarTriplesFromResource(o, p)$
7: **if** $correlations(s, p)$ is inconsistent **or** $correlations(o, p)$ is inconsistent **then**
8: $return \leftarrow$ add triple computed by correlation (s, p', o') or (o, p', o')
9: **end if**

3.2 Language Inconsistency

We define a language inconsistency in the database if, given more than one chapter in the database (more than one language instance for a database) there are two resources who, represent the same entity and are therefore considered the same, however, have no triple between them defined as being "owl:sameAs", which indicates they are the same resource.

One simple example of this would be if a given famous resource such as Mona Lisa was indeed present in all DBpedia chapters. For that resource to be fully language consistent all references to this resource must have between each of them a triple of the type (Resource in chapter A, owl:sameAs, Resource in chapter B) so that DBpedia knows they refer to the same resource. An example of that is shown in Figure 4.

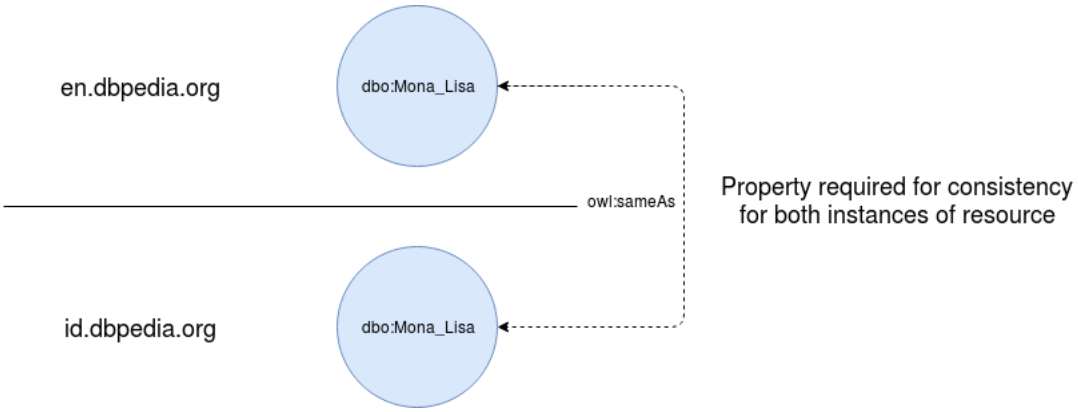

Figure 4: Example of link of owl:sameAs required between same resources in differet chapters for *DBpedia*.

Regarding the detection of our second defined inconsistency, the Language Incon-

sistency, we used the triples being inserted to verify for every chapter of the DBpedia if the given resource(s and o) had a representative in each language. First, we searched for all the triples with the property *owl:sameAs* and queried if they existed in each chapter. For every chapter that did not contain the resource, we would utilize the name of the resource to query if the resource did exist in the chapter, and simply did not have a triple referencing it as being the same as the original english one.

The method worked on it's own for resources that had proper names (names that simply are not translated in different languages). For all other resources, an automatic translation was employed using Google translator. However, the process greatly increased the runtime of the algorithm, making it unfit to execute concurrently with *DBpedia Live*. Therefore, we attained to make only simple queries with the names of the resources throughout the chapters of *DBpedia*. Algorithm 3 describes the process in further detail.

Algorithm 3: Language inconsistency detection algorithm.

Require: s, o
1: $sameAs(s) \leftarrow queryEveryTripleSameAsInResource(s)$
2: $referenced(s) \leftarrow queryResourceInChapters(s)$
3: **for all** $chapter \in referenced(s)$ **do**
4: **if** $chapter \notin sameAs$ **then**
5: s is language inconsistent in the language $chapter$
6: **end if**
7: **end for**
8: $sameAs(o) \leftarrow queryEveryTripleSameAsInResource(o)$
9: $referenced(o) \leftarrow queryResourceInChapters(o)$
10: **for all** $chapter \in referenced(o)$ **do**
11: **if** $chapter \notin sameAs$ **then**
12: o is language inconsistent in the language $chapter$
13: **end if**
14: **end for**

The Language Inconsistency correction algorithm was more straightforward when compared to the evolution one. Based on the resources that it detected to be the same in different chapters and only did not have a triple that defined them to be so, the method applied for correction was simply to add that triple. Unlike the Evolution case, here every resource detected was corrected.

The correction method is entirely based on the detection - if the detection algorithm identified a missing link between two resources then naturally the correct way of applying that correction is to add such missing link to the database.The method is illustrated in Algorithm 4, as is the same as the one used in detection, with the modification of lines 5 and 12 - the lines where we computed the inconsistency detection now add the missing link to *DBpedia*.

Algorithm 4: Language inconsistency correction algorithm.

Require: s, o

1: $sameAs(s) \leftarrow queryEveryTripleSameAsInResource(s)$
2: $referenced(s) \leftarrow queryResourceInChapters(s)$
3: **for all** $chapter \in referenced(s)$ **do**
4: **if** $chapter \notin sameAs(s)$ **then**
5: $return \leftarrow$ add triple $(s, owl : sameAs, resourceInChapter(s, chapter))$
6: **end if**
7: **end for**
8: $sameAs(o) \leftarrow queryEveryTripleSameAsInResource(o)$
9: $referenced(o) \leftarrow queryResourceInChapters(o)$
10: **for all** $chapter \in referenced(o)$ **do**
11: **if** $chapter \notin sameAs(o)$ **then**
12: $return \leftarrow$ add triple $(o, owl : sameAs, resourceInChapter(o, chapter))$
13: **end if**
14: **end for**

4 Evaluation

Having our procedures defined, we then aimed to analyze the effectiveness of our algorithms and understand the frequency each of those inconsistencies appeared. We also aimed to observe in the case of the Evolution Inconsistencies, how many could be corrected by our proposed solution of the triples detected.

4.1 Evolution Inconsistency

In the case of Evolution Inconsistency, the dataset utilized for the experiment was the RDF triple dump from "20190401" from the english chapter of the DBpedia. The dump was executed through the *DBpedia Live Extractor*, executing for the first 443765 triples. We utilized the *DBpedia Extraction Framework*, which is the same utilized in the DBpedia Live extraction, openly available[3]. We also utilized SPARQL queries via *SPARQLWrapper* in python to consult the DBpedia full database for other triples of interest.

In the first part of the experiment, we executed our detection algorithm to analyze, of all the triples added, how many were considered inconsistent. Then, we applied our correction algorithm over the same original dataset, and applied the detection algorithm on the result after correction to identify how many triples were still considered inconsistent. Table 1 presents the obtained.

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

Table 1: Results of the datasets with 443765 triples.

Original DBpedia Live Extraction		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Evolution Inconsistency	16132	3.63%
Extraction With our proposed Algorithm		
Type of inconsistency	Quantity of triples	Inconsistency Rate
Evolution Inconsistency	11497	2.59%

Table 1 demonstrates that there is not many occurrences of Evolution Inconsistency in a relative scope of the triples in *DBpedia*. However, due to the overall amount of triples that are added, there is a consistently large number of triples being considered inconsistent. The algorithm of detection performed well, considering it managed to detect 3.63% occurrences in our dataset, which is close to the expected value given our studies.

As for the correction algorithm, even though there could be improvement, as it only reduced the frequency of such inconsistency in about 1.04%, it caused over a quarter of inconsistent triples to become consistent again. The results may not be flashy, but they are a step in the right direction and prove that the correction can be automatically made and that the proposed solution is effective.

4.2 Language Inconsistency

As for the Language Inconsistency, the dataset utilized was the same - with the algorithm of correction executing queries over all chapters of *DBpedia* for every resource present. The queries over other chapters were also via *SPARQLWrapper* accessing the SPARQL endpoints for every different chapter.

Since the detection and correction methods work the same for this inconsistency type, we simply analysed how many triples with the property $owl : sameAs$ were added after the correction algorithm and studied over which chapters of DBpedia they were applied. The results are highlighted in table 2.

We can see that of all the thousands of triples, very few resources were actually detected to be inconsistent. Indeed, even with as many chapters as DBpedia contains, this is a more uncommon inconsistency. Still, it is present, and can cause problems with queries on the database, if you are looking for resources in multiple chapters.

The least populated chapters(such as indian) might have less resources, but the resources present are not as well defined in referencing their counterparts in other languages. This makes up for the amount of resources overall and indicate a more recurring presence of the inconsistency. Meanwhile chapters such as french, spanish

Table 2: Results of the datasets with 443765 triples.

Chapter of DBpedia	Number of resources with missing links found
English	8
Spanish	10
French	22
Italian	28
German	41
Portuguese	55
Indian	79
Total	243

and english are more well defined, contributing to less inconsistencies.

5 Discussion

This work aimed to improve the DBpedia consistency in an automatic way, by improving the consistency of all chapters of DBpedia and expanding on previous studies of inconsistencies.

When handling inconsistencies in *DBpedia*, and based on all the studies about DBpedia and RDF inconsistencies in general, it was expected to find the addition of triples that cause the dataset to become inconsistent. Having worked with other types of inconsistencies in a previous work [8], this investigation looked for different types of inconsistencies that were harder to identify.

As the results of our experiments show, the *DBpedia Live Extraction* is a tool that can make mistakes in the process of converting information from Wikipedia into RDF. Considering the effectiveness of having a tool update the *DBpedia on-the-fly*, we embraced the importance of making such a tool as reliable as possible.

Our algorithms developed a reliable cost-free way to improve the extraction currently utilized, that requires no human work other than code maintenance. The detection of triples can be applied to any other RDF repository that are lively updated and obey an ontology specification.

The proposed methods showed a relevant results to help future research to make an even better the *DBpedia Live Extraction* tool regarding the detection of different types of inconsistencies. Our improvements can be applied in any chapter of the DBpedia freely changing only the database on which the queries occur.

In addition, our study expands on multiple studies classifying all types of inconsistencies detected on RDF triple databases. Pushing forward the studies of different

types of inconsistencies is crucial for the advance of the Linked Data movement. Considering the continuous growth of the Web of Data databases like DBpedia will become more useful over time. With that, they will require to be more reliable.

To guarantee that reliability for *DBpedia* an improvement of the extraction tool is important. As the experiments showed, the original extraction generates many inconsistencies and the detection and further correction of those is very useful.

Some limitations, however, can be seen in the amount of triples corrected compared to the triples detected. While focusing only on automatic corrections, the number of decisions the algorithm can make while being certain that it won't create more inconsistencies is limited. It can improve slightly in efficiency in that regard by applying a deeper ontology study based on the property of every triple added.

What can limit the tool's effectiveness is the natural mistakes that occur when handling public generated information. Since the data added from *DBpedia* comes from Wikipedia, a free encyclopedia open for public edition, it is natural that some inconsistencies may occur. It is the job of the tool, however, to mitigate every possible mistake that can be generated by the conversion of information from the Wikipedia pages to RDF data, and we believe our tool has succeeded in that aspect.

Our tool could still be improved by tackling other types of inconsistencies. This could be more efficient by applying different correction methods aiming to correct 100% of all triples that have been discovered to be inconsistent. Nevertheless, we believe the work made an improvement to an ever growing database and made it more reliable.

6 Conclusion

DBpedia is getting bigger and more relevant with the continuous growth of Wikipedia. Due to this fact, it needs to be as reliable as ever. This work investigated how to improve the consistency of the *DBpedia* by defining two types of inconsistencies that could be present in any RDF database as well. Our investigation offered a solution on how to detect and correct them. We experimented on the *DBpedia Live Extraction* for applying our changes automatically as the data is added to *DBpedia*. Our experiments showed that the inconsistencies do occur in many triples added. The proposed tool turns *DBpedia* more reliable by correcting the identified inconsistent RDF triples. There is still room for improvements by considering the efficiency of correction and the possible types of inconsistencies. The proposed method is a step forward to the *DBpedia* reliability.

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] Sören Auer and Heinrich Herre. A versioning and evolution framework for rdf knowledge bases. In Irina Virbitskaite and Andrei Voronkov, editors, *Perspectives of Systems Informatics*, pages 55–69, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[3] 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.

[4] 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.

[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] Túlio Martins and Julio Cesar dos Reis. Mechanism for inconsistency correction in dbpedia live. Technical report, Universidade Estadual de Campinas - UNICAMP, 2019.

[9] 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.

[10] Saeedeh Shekarpour Sören Auer Maria-Esther Vidal Sidra Faisal, Kemele M. Endris. Co-evolution of rdf datasets. *Lecture Notes in Computer Science*, 9671, 2016.

[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. hal-00635854, 2011.