UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE MATEMÁTICA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

INGRID QUINTANILHA PACHECO

ETL4PROFILING: extending ETL4LOD to analyze datasets completeness – A DBpedia Case Study

RIO DE JANEIRO

2020

INGRID QUINTANILHA PACHECO

ETL4PROFILING: extending ETL4LOD to analyze datasets completeness – A DBpedia Case Study

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Maria Luiza Machado Campos

RIO DE JANEIRO

2020

CIP - Catalogação na Publicação

P116e

Pacheco, Ingrid Quintanilha
ETL4PROFILING: extending ETL4LOD to analyze datasets completeness - A DBpedia Case Study / Ingrid Quintanilha Pacheco. -- Rio de Janeiro, 2020.
101 f.

Orientadora: Maria Luiza Machado Campos.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2020.

1. DBpedia. 2. Dados abertos conectados. 3. Data Profiling. 4. ETL4LOD. 5. FAIR. I. Campos, Maria Luiza Machado , orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

INGRID QUINTANILHA PACHECO

ETL4PROFILING: extending ETL4LOD to analyze datasets completeness – A DBpedia Case Study

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 28 de Outubro de 2020.

To my family, who always supported
me in the hardest times.

ACKNOWLEDGEMENTS

I would like to thank my astonishing advisor Maria Luiza, who helped me in the elaboration of this project and encouraged me to proceed in my studies, even when all I had were doubts. Moreover, would like to thank Giseli Rabello Lopes for introducing me to InformAÇÃO, where I first had the chance to work with semantic web and felt in love for it. Thank you to Jean-Gabriel Nguema, for sharing his knowledge and guiding my way through Pentaho and DBpedia, becoming an excellent teacher to me. Thank you to every person who participated in this study. Thank you to God, for giving me hope when I almost gave up for thinking I wasn't good enough. Lastly, I would like to thank my family, my boyfriend and my friends, who kept me in place and cheered for me through my entire undergraduate course.

‘So, linked data -- it is huge. I've only told you a very small number of things There are data in every aspect of our lives, every aspect of work and pleasure, and it is not just about the number of places where data comes, it is about connecting it together. And when you connect data together, you get power in a way that doesn't happen just with the Web, with documents. You get this really huge power out of it.’

Tim Berners-Lee

RESUMO

À medida que a quantidade de dados no mundo cresce, é importante mantê-los acessíveis e usáveis, ao mesmo tempo que corretos e confiáveis. Além disso, o princípio R1 (Reuse)[1] da FAIR argumenta que é mais fácil encontrar e reusar dados se eles tiverem muitos rótulos atrelados a eles, considerando que ter uma boa qualidade de dados é essencial para qualquer repositório quando se trata de apoiar a sua abertura e reuso. Desta forma, o presente estudo tem a intenção de analisar as atuais condições de diversos conjuntos de dados, com um foco especial para a DBpedia, um projeto aberto que serve como um *hub* central na nuvem de dados conectados (*Linked Open Data Cloud*). Apesar de possuir mais de seis milhões de dados estruturados e seu grande uso para pesquisas e processos de aprendizado de máquina, ela contém muitos dados incompletos e recursos classificados erroneamente, o que dificulta a sua abertura e uso em projetos externos. A pesquisa é então baseada na extensão dos plugins ETL4LOD para análise de diferentes versões da DBpedia através de seus *templates*, fazendo uma caracterização ou perfil dos dados (*Data Profiling)* detalhado dos mesmos. Através dessa análise foi possível encontrar, dentre outras informações, a completude de 58.3% dos munícios brasileiros na DBpedia pt em comparação a 97.3% das cidades do Japão na DBpedia ja. Resumindo, apesar da DBpedia ser importante para os dados conectados, ela ainda apresenta dados incompletos, principalmente na versão portuguesa, que precisam ser trabalhados a fim de ajudar o repositório a se tornar mais completo e consequentemente apoiar o seu reuso em pesquisas e projetos futuros.

Palavras-chave: DBpedia. Dados Abertos Conectados. Data Profiling. ETL4LOD. FAIR.

[1] https://www.go-fair.org/fair-principles/r1-metadata-richly-described-plurality-accurate-relevant-attributes/

ABSTRACT

As the amount of data in the world increases, it is important to keep them accessible and usable, whereas accurate and trustworthy. Moreover, the FAIR principle R1 (Reuse) discusses that it is much easier to find and reuse data if there are many labels attached to them, considering that having a good data quality is essential to any repository when it comes to supporting their openness and reuse. Therefore, the present study has the intention of analyzing the current conditions of several datasets, with a special focus on DBpedia, an open source project that serves as a central hub in Linked Open Data Cloud. Although it has over six million structured data and its huge use for researches and processes of machine learning, it contains many incomplete data and wrong classified resources, which makes it difficult to open and use in external projects. Hence, the research is based on the extension of ETL4LOD plugins to analyze different versions of DBpedia through their templates, making a deeper Data Profiling of the mentioned dataset. From this analysis it was possible to find, among other information, the completeness of 58.3% from the Brazilian counties in DBpedia pt in comparison to 97.3% from the Japanese cities in DBpedia ja. To sum up, even though DBpedia is important to linked data, it still has incomplete data, especially in the portuguese version, that need to be treated aiming to help the repository to become more complete and consequently to support its reuse for future researches and projects.

Keywords: DBpedia. Linked Open data. Data Profiling. ETL4LOD. FAIR.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

BBC – British Broadcasting Corporation

COBOL – COmmon Business Oriented Language

CSV – Comma-separated Values

EOSC – European Open Science Cloud

ETL – Extract, Transform and Load

EU – European Union

FAIR – Findable, Accessible, Interoperable and Reusable

IBGE – Instituto Brasileiro de Geografia e Estatística

IBICT – Instituto Brasileiro de Informação em Ciência e Tecnologia

IBM – International Business Machines Corporation

ICICT – Institute of Communication and Scientific and Technological Information in Health

IFDS – Internet of FAIR Data and Services

IN – Implementation Network

IPUMS – Integrated Public Use Microdata Series

LOD – Linked Open Data

RDF – Resource Description Framework

SPARQL – SPARQL Protocol and RDF Query Language

TXT – Text

URI - Universal Resource Identifier

URL – Uniform Resource Locator

TABLE OF CONTENTS

1 INTRODUCTION

It is unquestionable that humanity is becoming more digitally influenced as time progresses. Daily, an enormous volume of information is produced, transferred, and saved in computers and equipments of all kinds around the world. Among the circulating data, those motivating this work are known as Open Data.

The second decade of the 2000s was marked by the exponential growth of the digital sector and the management of virtual information. According to the International Data Corporation (2012), Big Data enterprises are on the rise since 2012, demonstrating the potential and value of the mentioned sector. Moreover, whereas the digital universe held 1227 exabytes back in 2010, it is estimated to reach 40000 exabytes by the end of the year 2020, all due to the burst in the amount of circulating data in the world.

A way to understand the emergence of an abundance of data is by analyzing its definition. Regarding the concept of data, the Merriam-Webster dictionary (2019) discusses that it is any and all information that holds any use and, in this case, that shows itself in the digital format, capable of being processed and spread. Therefore, it is understandable that the growth is due to the fact that never before so many digital resources have been captured and stored. For instance, cameras, microphones, and sensors of all kinds generate and collect data, while sites and companies create registers and files that are sent everywhere.

Once defined what is data and its value to the contemporaneous society, the focus moves back to open data, the specific type of data addressed in this work.

According to the definition presented in the Portal Brasileiro de Dados Abertos[2], "data is considered open when anyone can freely access it, use it, modify it and share it for any purpose, being subjected to the requirements that aims the preservation of provenance and openness."

This way, it is comprehensible the meaningful value open data has to society. As discussed by the Open Knowledge Portal[3], it promotes transparency, a fundamental principle to the association between citizens and the government, boosts the creation of new business, as

[2] http://dados.gov.br/pagina/dados-abertos#:~:text=dados%20s%C3%A3o%20abertos%20quando%20qualquer,sua%20proveni%C3%AAncia%20e%20sua%20abertura.&text=Os%20dados%20abertos%20tamb%C3%A9m%20s%C3%A3o%20pautados%20pelas%20tr%C3%AAs%20leis%20e%20oito%20princ%C3%ADpios.

[3] https://okfn.org/opendata/why-open-data/

all kinds of business need information to support their decision processes, and, finally, foster greater engagement between organizations and the public.

1.1 MOTIVATION

The Open Data Barometer[4], a global measurement created by the World Wide Web Foundation with the support of the Omidyar Network, stated, in a survey conducted in 2016 with 115 countries and jurisdictions, that nine out of ten government datasets were not open, a decrease from 10 to 7% in a year in the amount of data that were fully open, and likewise a decrease in those that were machine-readable.

Moreover, less than 31% of the published datasets discovered in the research had basic metadata or some supportive documentation. These information help to enlighten some major issues considering open data: they are either not open to the public, incomplete or in formats that hamper their usage in a range of applications.

One of the cases in Brazil, for instance, is the so-called Dirty List of Slave Labor, released by the Brazilian Government on some occasions in limited media formats that make it difficult to process or to have information extracted from them.

Intending to mitigate these adversities, the GO FAIR[6] initiative emerged in 2017 in Europe. It aims to implement a global internet of FAIR services and data worldwide, following a set of principles to guarantee that data are findable, accessible, interoperable, and reusable (WILKINSON, Mark D. et al., 2016), using metadata as their primary tool of assessment.

Furthermore, there is a supporting process to generate FAIR data and metadata called FAIRification (JACOBSEN, Annika et al., 2020). It has been so widely recognized that many institutions in Brazil have been discussing their strategies to evolve research data according to FAIR principles, for instance, the GO FAIR Brazil Network and the GO FAIR Brazil Health Network, coordinated by the Brazilian Institute of Information in Science and Technology (IBICT) and Oswaldo Cruz Foundation (FIOCRUZ), respectively.

FAIR principles focus on openness, reuse and accessibility of data, therefore, sharing new data with a good quality is essential, especially when it comes to research material.

One of the groups dedicated to reverse the situation, by opening, cataloging, and facilitating access to data is DBpedia, an open source project founded between 2006 and 2007

[4] https://opendatabarometer.org/4thedition/report/#findings_recommendations
[6] https://www.go-fair.org/

to create and provide public access to structured data, supporting queries and searches over its properties and relationships. It serves as a central hub in the Linked Open Data Cloud, by extracting Wikipedia data and allowing them to be related to other databases.

However, despite being an extremely important initiative due to its value for researches and processes of machine learning, natural language processing and the semantic Web, the base is neither stable in terms of consistency nor in terms of completeness of its published data. Some identified problems include inconsistencies of categories, redundancies and erroneous conceptualization of some resources.

1.2 OBJECTIVE

Since the main objective of this study is to help datasets become more reliable for storing scientific data, it focusses on punctuating their critical points, providing a detailed report about the findings. Moreover, it uses DBpedia as the main case study due to its tremendous contribution to the LOD Cloud.

A data profiling process allows the user to more easily find the main spots that need attention in a dataset and to know where to start improving its quality. It also helps the user to provide more accurate data when submitting them to any transformation process, as, for example, to generate and annotate linked data, checking the accuracy and completeness before inserting them in a triple store or a repository. That way, more universities and institutions will feel safe to use them to store their research results and make their connections richer, encouraging the opening of data and facilitating their reuse.

To sum up, the goals this thesis aims to stand out about the datasets are:

- Point out their required improvements;
- Return statistics about their current data;
- Find correlation between different references concerning the same data;
- Compare distinct versions inside them to know what is missing in each;

1.3 CONTRIBUTION

The primary contribution was a framework called ETL4Profiling, based on the ETL (Extract, Transform and Load) workflow. It was implemented on Kettle (Pentaho Data Integration[8]) and had the intention of analyzing diverse datasets in terms of their completeness.

The set has 8 plugins focused on evaluating DBpedia data and other 2 considered to be the first steps to a general evaluation of datasets. While the first gets and compares data using the template properties and resources as references, the second uses the datasets models to perform the comparisons.

Later, to confirm their performances two experiments were created, putting the plugins to test in a practical approach. First, two versions of DBpedia were compared, Portuguese and Japanese, finding their correlations. As a result, DBpedia pt was slightly worse than the Japanese when it comes to each resource/property quality, but better regarding the template completeness.

At last, the Harvard and Open DataSUS datasets were compared, enlightening the missing properties from the first with reference to the second, leading to a completeness of 60%.

1.4 STRUCTURE

This introduction included basic concepts and topics about the area, such as semantic Web, LOD, open data, DBpedia and the FAIR principles, as well as the motivation behind this work and the implemented solution: ETL4Profiling.

In Chapter 2, a contextualization of the literature used in this project is presented, by showing the most important concepts and explaining them in details.

Chapter 3 goes deeper into the technicalities of the proposed plugins, describing more about the initial steps of the process and even the specifications of each.

Chapter 4 exemplifies the usage of each plugin in real applications, showing the results obtained when applying them in DBpedia and some COVID-19 related datasets.

Finally, in Chapter 5, a conclusion is presented, including major discoveries, difficulties found, and future work.

[8] https://community.hitachivantara.com/s/article/data-integration-kettle

2 LITERATURE REVIEW

Because of the expansion of open data, there is a rising concern about the data quality around the world. Some initiatives are gaining weight and investigating how to improve what already exists and what is being created.

First of all, it is important to review and analyze some concepts and models that deal with these issues and how they are evolving.

2.1 SEMANTIC WEB AND LINKED DATA

2.1.1 Basic Concepts

It comes as no surprise that, over time, the amount of data available on the Web has grown enormously. Although this is positive due to the wide variety of information it provides, it ends up becoming meaningless if some attention to their quality is not taken, considering that the Web's current data is designed for humans to read, not computers, as mentioned by Tim Berners-Lee, Hendler and Lassila in the article "The Semantic Web" (2001).

The Semantic Web gives meaning for the contents on the Web, in a way that software agents can read, understand and work on them, carrying out sophisticated tasks for users. Moreover, it is important to mention that the Semantic Web is an extension of the Web, in which meaning is attributed to information, allowing humans and computers to work together.

According to Ismail and Shaikh (2016), the Semantic Web tries to classify its data based on different topics and associate meaning to them, aiming to help both in human and machine understanding. Furthermore, as Hendler, Berners-Lee and Miller (2002) mention, it is based on the idea of having a Web where data is available, interlinked and well-defined in a way that they can be discovered and used by other software, reaching the World Wide Web's universal potential.

Besides, the *World Wide Web Consortium* (W3C)[9] specifies that they are helping in the development of technologies to support the "Web of data", data that you find in databases. For W3C, Semantic Web refers to their vision of linked data, which will be discussed in the following part. In addition, some technologies were created aiming to enable people to create

[9] https://www.w3.org/standards/semanticweb/

data stores on the Web and build vocabularies, such as RDF, SPARQL, and OWL (W3C, 2015?), which will also be explained in the next sections.

2.1.2 Resource Description Framework - RDF

RDF is a data interchange specification that occurs on the Web (a standard model), created by W3C (W3C, 2014). It is used as a method for conceptual description or information modeling but does not specify what that information means. As RDF allows data to fit in a determined pattern (standard), it can be easily interconnected to other data, making it simple to integrate them. Moreover, any kind of data, regardless of format, can be converted into RDF.

The first release was in 2004 by W3C and was called RDF 1.0. The newest version was published in 2004, the RDF 1.1.

It forms a labeled graph, where the edges represent the link between two resources. This uniform structure connects all the data through triplets (subject, predicate, and object/literal) as shown in figure 1.

In the statement "The Mona Lisa was created by Leonardo Da Vinci" the subject is “The Mona Lisa”. The relationship she has, that is, the predicate, is that she was created by someone, and the object is Leonardo da Vinci, which is another subject (figure 2). Objects can be literal values, such as birthdates, or even other subjects. This structure makes it easy to navigate between different data and find their metadata through queries.

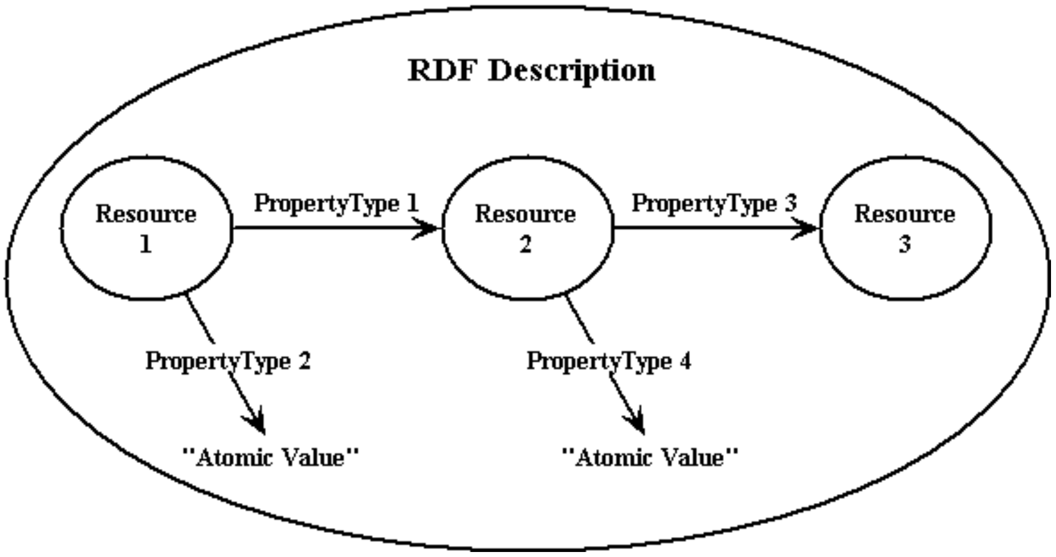


Figure 1 - RDF Structure. Source: https://medium.com/@atakanguney94/introduction-to-resource-description-framework-and-sparql-rdf-101-5857f4a6a8a6

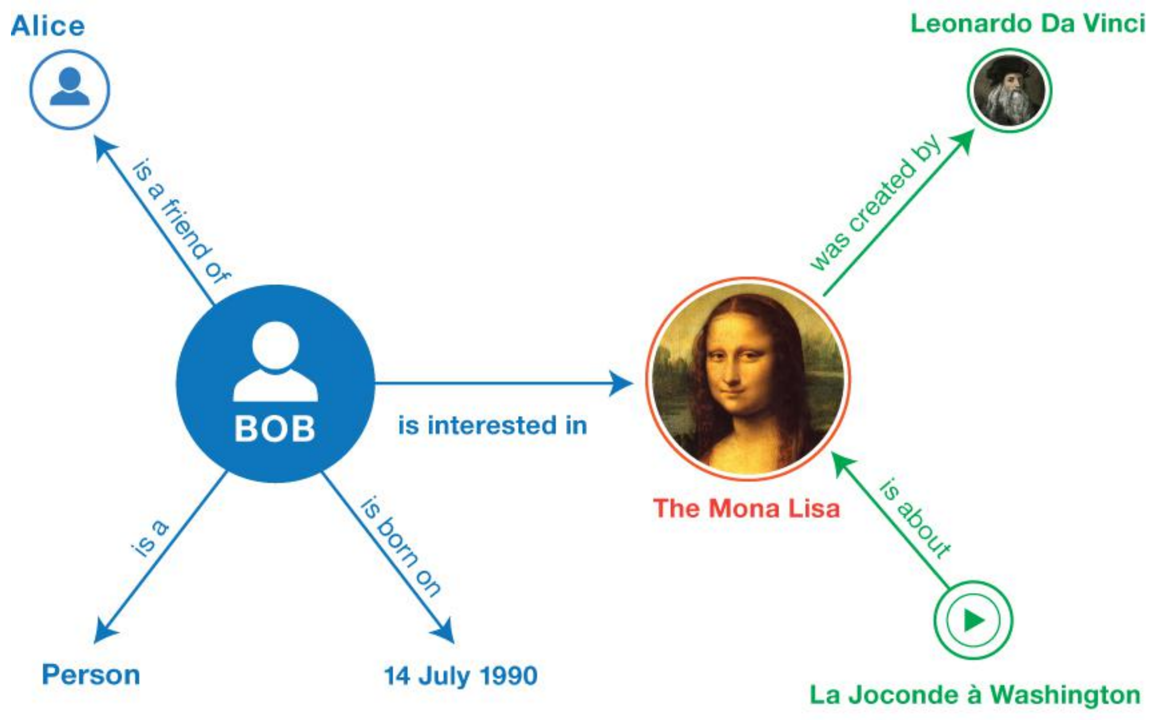


Figure 2 - Informal graph of the same triples. Source: https://dvcs.w3.org/hg/rdf/raw-file/tip/rdf-primer/Overview.html

Taking a better look at each part of the triple, three types of RDF data can appear: IRIs, literals and blank nodes (WORLD WIDE WEB CONSORTIUM et al., 2014).

IRI is short for "International Resource Identifier", which is a generalization of URI (Uniform Resource Identifier). It can appear in all possible positions of the triple and can be used to identify both documents and things, for instance, *http://dbpedia.org/resource/Leonardo_da_Vinci* is Leonardo da Vinci's IRI on DBpedia.

Literals are, as the name describes, literal values. They can be either strings, dates, or numbers, that do not have any other connection. They can only appear in the object position and can optionally be associated with a *language tag*, for example, "Léonard de Vince"@fr.

Lastly, blank nodes are used to mention resources without having to use an identifier. It can be placed in the subject or object positions and, briefly, represent something without saying its value.

There are several different serialization formats of RDF. Some of them are Turtle, N-triples (used to triplify DBpedia), N-Quads, JSON-LD, Notation3, RDF/XML, and RDF/JSON.

Taking into consideration the structure in figure 3 it would be possible to create a SPARQL query to find people who are interested in paintings by Leonardo Da Vinci, which would return Bob as the answer.

```
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<The Mona Lisa> <was created by> <Leonardo da Vinci>.
<The video 'La Joconde à Washington'> <is about> <the Mona Lisa>
```

Figure 3 - Example of an RDF structure with triples. Source: https://dvcs.w3.org/hg/rdf/raw-file/tip/rdf-primer/Overview.html

2.1.3 Vocabularies and ontologies

Both vocabularies and ontologies, in the Semantic Web, give meaning to resources, but in a different context. They are correlated concepts that aggregate different connotation to their terms.

In addition, vocabularies are computer-readable and have URIs that match terms and descriptions, while ontologies are more than just a vocabulary, with hierarchies and relations between concepts, representing a domain conceptualization (TAYE, Mohammad M., 2010). This way, it is possible to find and classify data by their terms and concepts.

2.1.3.1 Vocabularies

Vocabularies are well-defined terms used in communication (figure 4). They should not be redundant, and if so, not without explicitly declaring and identifying the redundancy. Moreover, they are expected to have consistent meaning in all contexts. (HEBELER, John et al., 2009, p. 99).

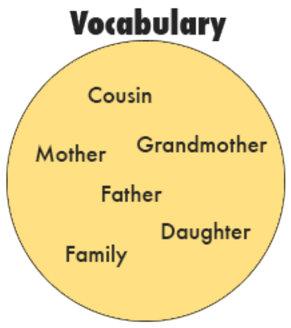


Figure 4 - Vocabulary as a simple collection of well-defined terms

It is important to note that RDF triples on their own do not capture the meaning of the information and, therefore, a vocabulary describing the classes of resources consistently is required.

2.1.3.2 Ontologies

Ontologies use vocabularies' terms to define concepts and relations between them (HEBELER, John et al., 2009, p. 100). They are usually mentioned as a well-defined model (logic-based) for describing a knowledge domain.

In its oldest definition, Ontology comes from a branch in philosophy that studies the nature of existence and the structure of reality (JACOB, 2003). Taking into consideration the Semantic Web field, ontology would identify the partial conceptualization of a shared given knowledge domain (JACOB, 2003).

Ontology, then, describes the domain, properties, and relations between concepts, becoming fuller and more expressive than vocabularies.

On top of that, there are plenty of relevant ontologies on the Web, being DBpedia's, one of them. It contains more than 5 million resources just in its English version, divided into classes, and described by different properties. This information confirms its relevance for this research, the reason why it became its main case study.

2.1.4 SPARQL Protocol and RDF Query Language - SPARQL

In the Semantic Web, SPARQL is the language used to search and manipulate RDF data in the Web of Data.

SPARQL queries are based on triples (figure 5). They provide some searches inside the RDF and return all triples that match them.

```
SELECT DISTINCT ?uri ?nome
  WHERE {
    ?uri dbo:nationality dbr:Brazil.
    ?uri dbo:occupation dbr:Politician.
    ?uri rdfs:label ?label.
    FILTER langMatches(lang(?label),"pt").
    BIND (STR(?label) AS ?nome)
  }
```

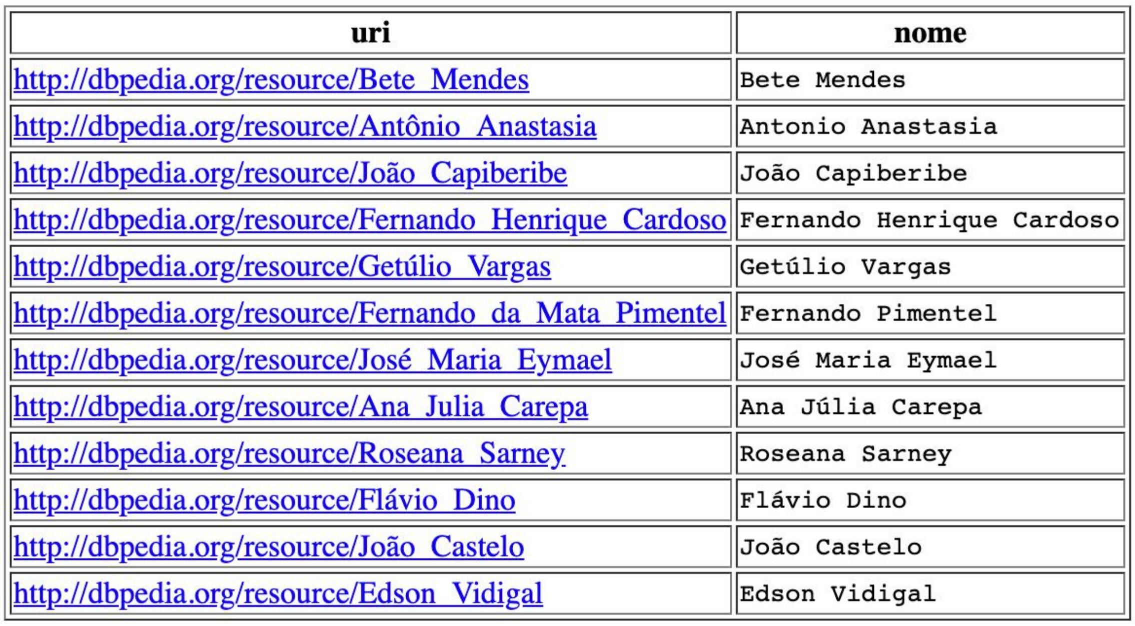
Figure 5 - SPARQL query running on DBpedia

Regarding how the language works, it consists of a simple structure with two clauses: SELECT and WHERE. SELECT identifies what is going to appear in the results and WHERE is the pattern that should match the triples.

In the following example, the SPARQL query is running on DBpedia[13], aiming to get all resources with occupation "politician" that have Brazil as nationality and that have a

[13] http://dbpedia.org

Portuguese label (FILTER parameter). It, then, as a result, returns both the URI and the name of the resources found (figure 6).



uri	nome
http://dbpedia.org/resource/Bete_Mendes	Bete Mendes
http://dbpedia.org/resource/Antônio_Anastasia	Antonio Anastasia
http://dbpedia.org/resource/João_Capiberibe	João Capiberibe
http://dbpedia.org/resource/Fernando_Henrique_Cardoso	Fernando Henrique Cardoso
http://dbpedia.org/resource/Getúlio_Vargas	Getúlio Vargas
http://dbpedia.org/resource/Fernando_da_Mata_Pimentel	Fernando Pimentel
http://dbpedia.org/resource/José_Maria_Eymael	José Maria Eymael
http://dbpedia.org/resource/Ana_Julia_Carepa	Ana Júlia Carepa
http://dbpedia.org/resource/Roseana_Sarney	Roseana Sarney
http://dbpedia.org/resource/Flávio_Dino	Flávio Dino
http://dbpedia.org/resource/João_Castelo	João Castelo
http://dbpedia.org/resource/Edson_Vidigal	Edson Vidigal

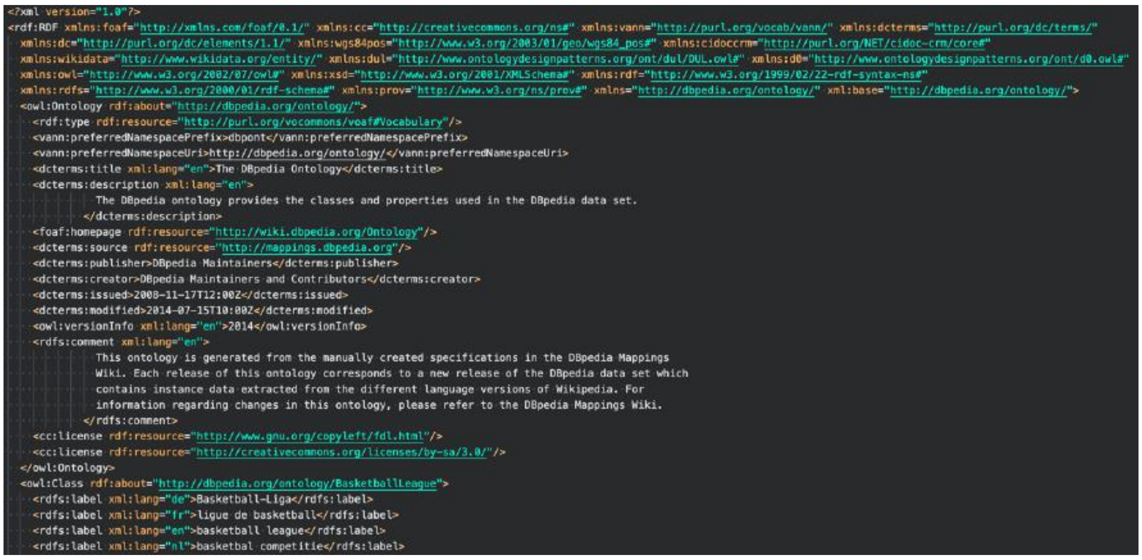
Figure 6 - Query's result

2.1.5 Ontology Web Language - OWL

The Ontology Web Language (OWL) is a Semantic Web language created to define complex and complete ontologies (W3C, 2012). It is a language understood by computers, in a way they can verify the consistency of the knowledge. Their documents, called ontologies, can be referred and refer to other OWL ontologies.

It became a W3C recommendation in 2004, after three years invested in its research and development (HEBELER, John et al., 2009). Later, it had its second version, known as OWL 2, published in 2009, with a Second Edition in 2012.

As OWL is based on RDF, there is no difference between the ontology and the data it describes. They contain three main semantic building blocks: classes (a set of resources), individuals (any resource inside a class), and properties (used to describe a resource) (HEBELER, John et al., 2009). An example to illustrate is DBpedia ontology in figure 7. Furthermore, they can optionally contain headers, that describe the resources. Headers usually contain comments, labels, information about the version, and import statements.



```
<?xml version="1.0"?>
<rdf:RDF xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:cc="http://creativecommons.org/ns#" xmlns:vann="http://purl.org/vocab/vann/" xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:wgs84pos="http://www.w3.org/2003/01/geo/wgs84_pos#" xmlns:cidoccrm="http://purl.org/NET/cidoc-crm/core#"
  xmlns:wikidata="http://www.wikidata.org/entity/" xmlns:dul="http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#" xmlns:d0="http://www.ontologydesignpatterns.org/ont/d0.owl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:prov="http://www.w3.org/ns/prov#" xmlns="http://dbpedia.org/ontology/" xml:base="http://dbpedia.org/ontology/">
  <owl:Ontology rdf:about="http://dbpedia.org/ontology/">
    <rdf:type rdf:resource="http://purl.org/vocommons/voaf#Vocabulary"/>
    <vann:preferredNamespacePrefix>dbpont</vann:preferredNamespacePrefix>
    <vann:preferredNamespaceUri>http://dbpedia.org/ontology/</vann:preferredNamespaceUri>
    <dcterms:title xml:lang="en">The DBpedia Ontology</dcterms:title>
    <dcterms:description xml:lang="en">
            The DBpedia ontology provides the classes and properties used in the DBpedia data set.
          </dcterms:description>
    <foaf:homepage rdf:resource="http://wiki.dbpedia.org/Ontology"/>
    <dcterms:source rdf:resource="http://mappings.dbpedia.org"/>
    <dcterms:publisher>DBpedia Maintainers</dcterms:publisher>
    <dcterms:creator>DBpedia Maintainers and Contributors</dcterms:creator>
    <dcterms:issued>2008-11-17T12:00Z</dcterms:issued>
    <dcterms:modified>2014-07-15T10:00Z</dcterms:modified>
    <owl:versionInfo xml:lang="en">2014</owl:versionInfo>
    <rdfs:comment xml:lang="en">
            This ontology is generated from the manually created specifications in the DBpedia Mappings
            Wiki. Each release of this ontology corresponds to a new release of the DBpedia data set which
            contains instance data extracted from the different language versions of Wikipedia. For
            information regarding changes in this ontology, please refer to the DBpedia Mappings Wiki.
          </rdfs:comment>
    <cc:license rdf:resource="http://www.gnu.org/copyleft/fdl.html"/>
    <cc:license rdf:resource="http://creativecommons.org/licenses/by-sa/3.0/"/>
  </owl:Ontology>
  <owl:Class rdf:about="http://dbpedia.org/ontology/BasketballLeague">
    <rdfs:label xml:lang="de">Basketball-Liga</rdfs:label>
    <rdfs:label xml:lang="fr">ligue de basketball</rdfs:label>
    <rdfs:label xml:lang="en">basketball league</rdfs:label>
    <rdfs:label xml:lang="nl">basketbal competitie</rdfs:label>
```

Figure 7 - DBpedia ontology

Additionally, OWL's vocabulary is defined in the namespace *http://www.w3.org/2002/07/owl*, identified by the prefix owl. Concerning its newest version, the namespace was maintained.

2.1.6 Linked Data

Despite the indisputable benefits the Web brought, it is a bit surprising that the same principles used for its growth were not applied to data until a decade ago. (BERNERS-LEE et al., 2009). Before that, data published on the Web were only available in raw formats, such as CSV, XML, or even HTML tables, making it harder to read and understand its semantic and structure.

It is important to note that as being the most concrete application of the Web of Data, linked data refer to published data in a way they can be machine-readable, have a defined meaning, and are linked to external datasets, even having the possibility that other datasets are linked to them as well. As Tim Berners-Lee mentions in his TED speech in 2009:

> Data is relationships. Interestingly, data is relationships. This person was born in Berlin; Berlin is in Germany. And when it has relationships, whenever it expresses a relationship then the other thing that it's related to is given one of those names that starts HTTP. So, I can go ahead and look that thing up. So I look up a person -- I can look up then the city where they were born; then I can look up the region it's in, and the town it's in, and the population of it, and so on. So I can browse this stuff. (BERNERS-LEE, Tim, 2009)

Unlike the Web of hypertext, where relations between documents are built from hypertext documents written in HTML, in linked data they create links between things described by RDF. Moreover, there are four main rules data must follow to ensure their sharing and data interconnection (BERNERS-LEE, 2009):

1. Use URIs as names (identifier);
2. Use HTTP URIs so that people can search these resources with their identifiers;
3. Provide useful information when someone looks up an URI, using RDF and SPARQL standards;
4. Include links to other URIs, to enable the discovery of related resources.

Unfortunately, even though some organizations have made their data available since the publication of linked data, many of them did not have a good quality. With this mindset, in 2010, Tim Berners-Lee developed the 5-star deployment scheme for open data, shown in figure 8 and described in table 1.

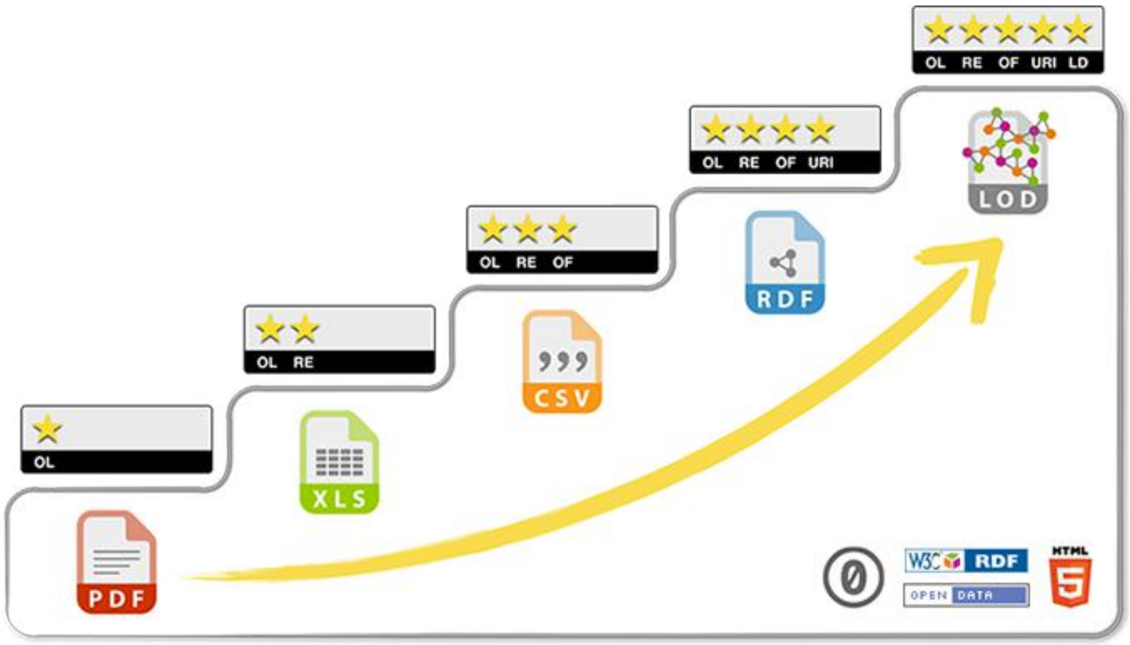


Figure 8 - 5-star deployment scheme for Open Data. Source: https://www.w3.org/DesignIssues/LinkedData.html

Table 1: 5-star deployment scheme description

LEVEL	DESCRIPTION
☆	Available on the Web in any format under an open license
☆☆	Available as structured data (machine-readable)
☆☆☆	Available in a non-property open format (CSV)
☆☆☆☆	Use URI to describe resources (open standards from W3C), so that people can point at them

☆☆☆☆☆	All the previous + Link data to other's people data (to give context)

Source: 5 Star Open Data website[15]

As seen in figure 8, the best format (5 stars) is Linked Open Data (LOD). LOD is linked data that is published under an open license, making it possible to be reused freely.

The biggest representation of LOD is the Linked Open Data Cloud[16], that shows datasets that were published following the linked data principles. It contained, in May 2020, 1260 datasets with 16187 links. Furthermore, it stores all released versions since 2007, when it had only 12 datasets.

In addition, the LOD Cloud maintainers allow users to contribute to the diagram by adding a new dataset to the cloud. The users datasets must follow both the linked data and the cloud principles:

1. They must begin with http:// or https://;
2. They must resolve, with or without content negotiation, to one of the RDF formats;
3. They must contain at least 1000 triples;
4. They must be connected through RDF links to a dataset that is already in the LOD Cloud, and must contain at least 50 links;
5. Their access must be viable via RDF crawling, RDF dump or SPARQL endpoint;

[15] https://5stardata.info/en/
[16] https://lod-cloud.net/#about

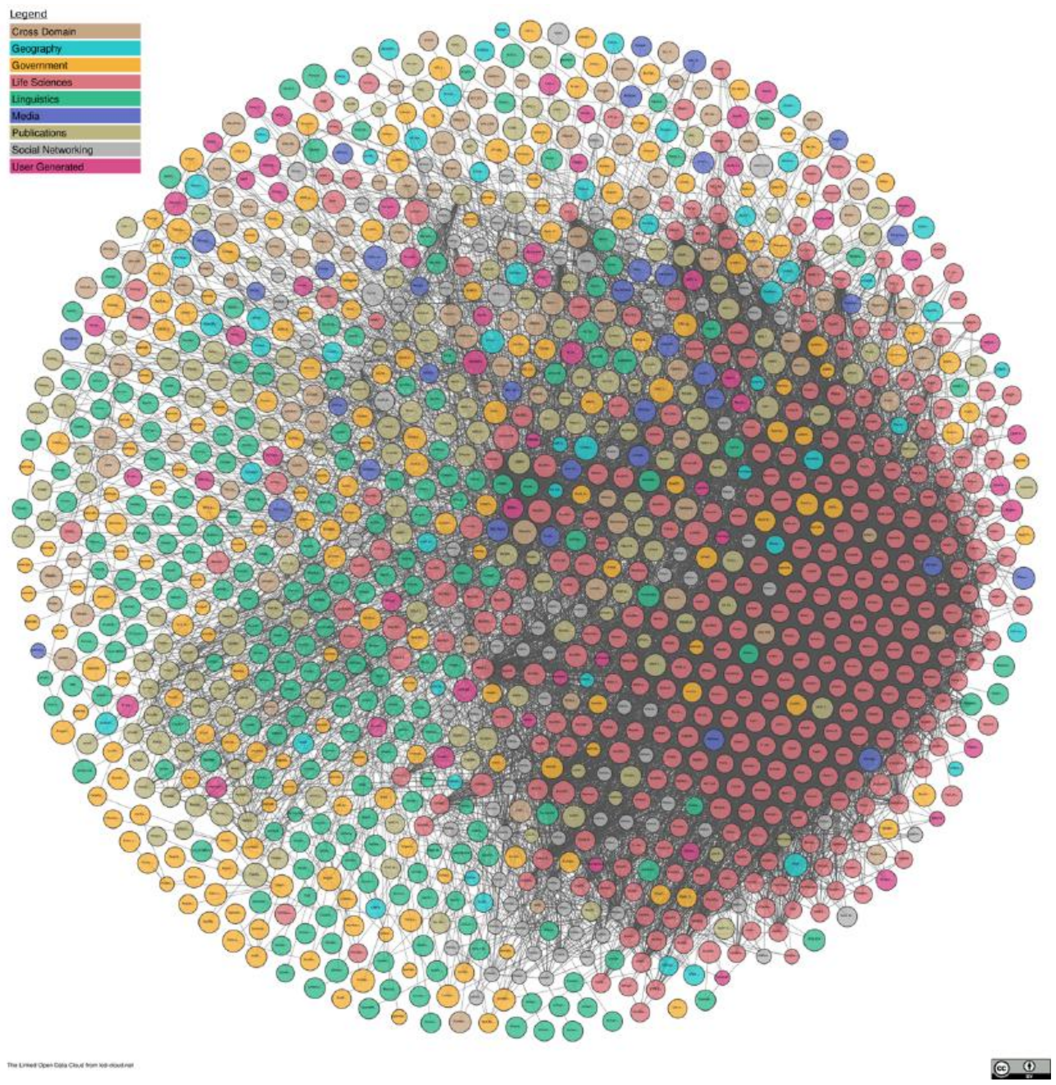


Figure 9 - LOD Cloud updated version. Source: https://lod-cloud.net/versions/2020-07-27/lod-cloud.png

In the newest version of the LOD Cloud (figure 9) it is clear the abundance of datasets and links. Plenty of them are linked in tangled connections, but there is one repository that stands out, the main hub, DBpedia.

2.2 DBPEDIA STRUCTURE

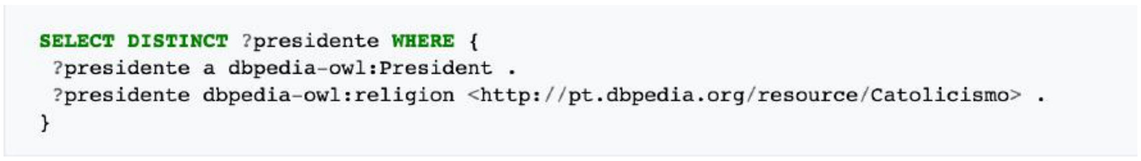
DBpedia began as a project by researchers from Mannheim and Leipzig universities[18]. They already had the idea of a free, open, and reliable data repository. Thus, they planned to extract structured data from Wikipedia, since it does not only contain free text but also structured data in infoboxes, with tables, lists, and others.

[18] https://wiki.dbpedia.org/about/dbpedia-community

In July 2020[19], Wikipedia was the 14th most popular website according to alexa.com, known worldwide and a big example of collaborative content. It has more than 200 editions in different languages that range in size from hundreds to millions of articles, as the English version, its widest version (MORSEY, Mohamed et al., 2012).

Although the amount of data is huge, its search engine is not extremely developed and finding an article, or a specific page may be difficult. As it only has textual search, it is hard to find more structured or complex data, for instance, the list of countries that have more than 100 million inhabitants.

DBpedia, otherwise, provides a way to find this, structuring the data extracted from Wikipedia, and, hence, being able to answer more expressive queries, such as the list of catholic presidents (figure 10).



```
SELECT DISTINCT ?presidente WHERE {
 ?presidente a dbpedia-owl:President .
 ?presidente dbpedia-owl:religion <http://pt.dbpedia.org/resource/Catolicismo> .
}
```

Figure 10 - SPARQL query that searches all presidents that are catholic in the Portuguese DBpedia. Source: http://pt.dbpedia.org/exemplos/

It was released to the public in 2007, allowing users to query linkages between various Wikipedia pages. DBpedia would soon become one of the most famous and biggest parts of the decentralized linked data effort, as mentioned by Tim Berners Lee in a TED Talk in 2009.

> So he wrote a program to take the data, extract it from Wikipedia, and put it into a blob of linked data on the Web, which he called DBpedia. DBpedia is represented by the blue blob in the middle of this slide and if you actually go and look up Berlin, you'll find that there are other blobs of data which also have stuff about Berlin, and they are linked together. So if you pull the data from DBpedia about Berlin, you'll end up pulling up these other things as well. And the exciting thing is it is starting to grow. This is just the grassroots stuff again, OK? (BERNERS-LEE, Tim, 2009)

The development of DBpedia began years before 2007. At that time, the developers were trying to create links between different datasets and isolated data throughout the internet. This project, born from a collaboration between the two universities previously mentioned, was able to extract the important information from Wikipedia and turn it into a linked data repository, making this data accessible and usable on the internet.

[19] https://www.alexa.com/topsites

Later, it made possible querying and linking data extracted from Wikipedia, changing the way they were provided. Their main goal was not only to get this not originally accessible (usable) data, but to provide them openly to the world so that they could be used in researches and other projects as well.

To understand its context, an analysis of the data available on the internet at the time is crucial. In 2007, the year DBpedia was released, Wikipedia, as well as other repositories, was fairly young. By the time, little attention was taken about how data were provided or if they had associations with others. Therefore, DBpedia emerged at the most appropriate time, with the fast expansion of the Internet in the middle of the years 2000 and the increasing interest in Big Data in 2005.

Nowadays, with its growth, not only large Semantic Web research communities use it but also big companies such as BBC and New York Times, to organize their content.

DBpedia volume of data and quality increased so remarkably that it became the linking hub in the Web of linked data, setting RDF links to a great extent of external data sources.

To reach this number, DBpedia extracted data directly from Wikipedia, through an extraction framework structured in four parts. It extracts Wikipedia structured information and transforms them into a knowledge base as shown in figure 11.

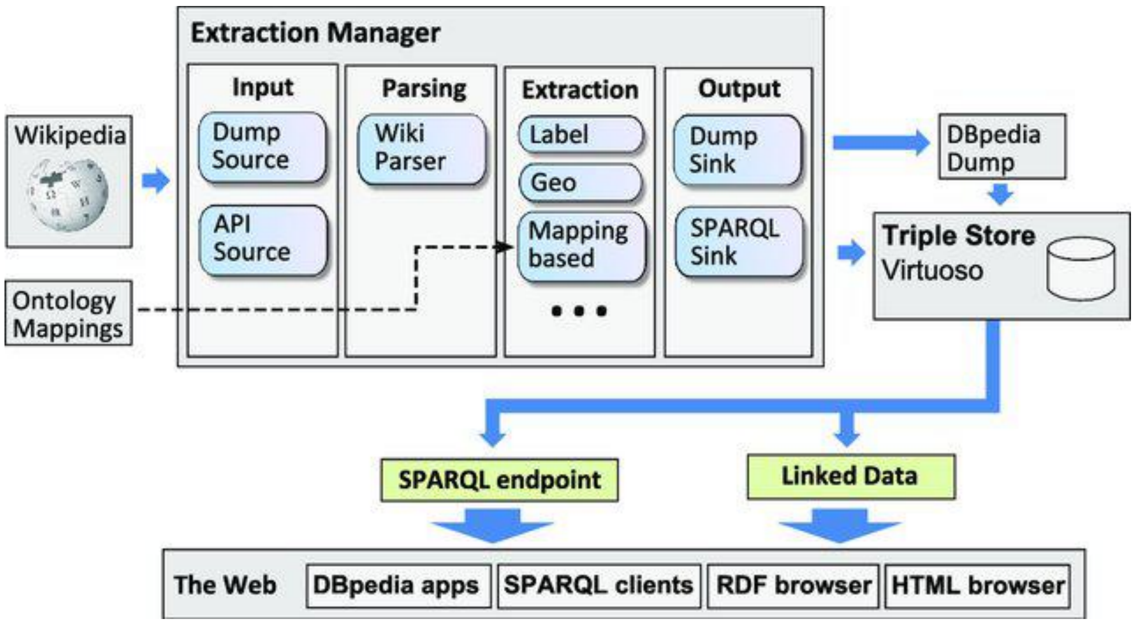
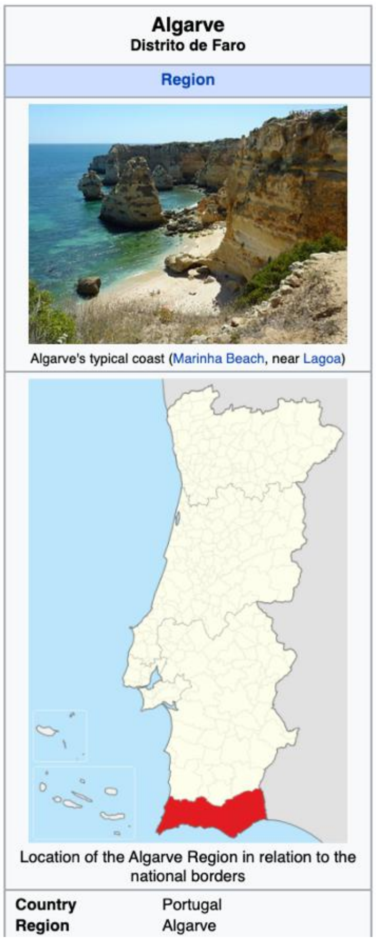


Figure 11 - DBpedia extraction framework. Source: http://jens-lehmann.org/files/2015/swj_dbpedia.pdf

First, the Wikipedia pages, more specifically the infoboxes (figure 12), are read from an external source. They can either be read from a Wikipedia dump or fetched from the MediaWiki API (figure 13) (LEHMANN, Jens et al., 2015).



Algarve
Distrito de Faro

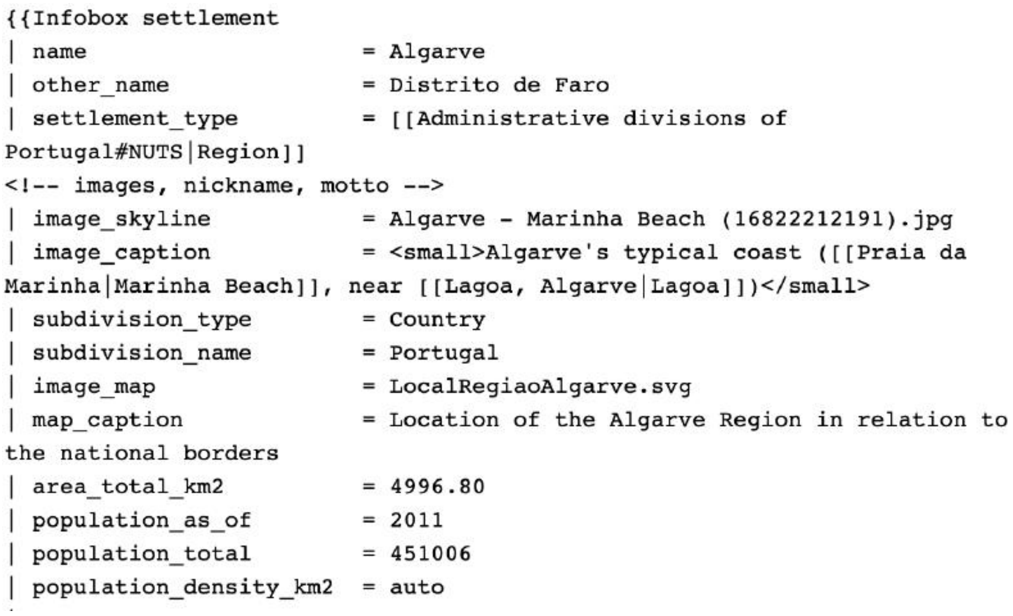
Region

Algarve's typical coast (Marinha Beach, near Lagoa)

Location of the Algarve Region in relation to the national borders

Country	Portugal
Region	Algarve

Figure 12 - Algarve Wikipedia infobox. Source: https://en.wikipedia.org/wiki/Algarve



```
{{Infobox settlement
| name                     = Algarve
| other_name               = Distrito de Faro
| settlement_type          = [[Administrative divisions of
Portugal#NUTS|Region]]
<!-- images, nickname, motto -->
| image_skyline            = Algarve - Marinha Beach (16822212191).jpg
| image_caption            = <small>Algarve's typical coast ([[Praia da
Marinha|Marinha Beach]], near [[Lagoa, Algarve|Lagoa]])</small>
| subdivision_type         = Country
| subdivision_name         = Portugal
| image_map                = LocalRegiaoAlgarve.svg
| map_caption              = Location of the Algarve Region in relation to
the national borders
| area_total_km2           = 4996.80
| population_as_of         = 2011
| population_total         = 451006
| population_density_km2   = auto
```

Figure 13 - Algarve Wikipedia Source Code. Source: https://en.wikipedia.org/w/index.php?title=Algarve&action=edit

Then, they are parsed by the wiki parser, that transforms their source codes into an Abstract Syntax Tree, which is a tree representation of the abstract syntactic of the source code (figure 14).

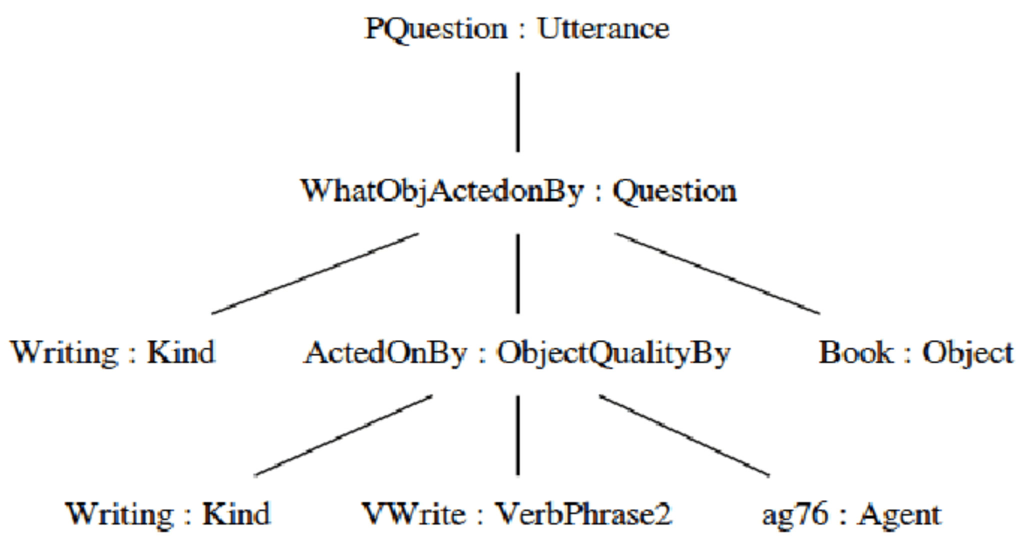


Figure 14 - Example of Abstract Syntax Tree. Source: https://www.researchgate.net/publication/263218121_Romanian2SPARQL_A_Grammatical_Framework_approach_for_querying_Linked_Data_in_Romanian_language

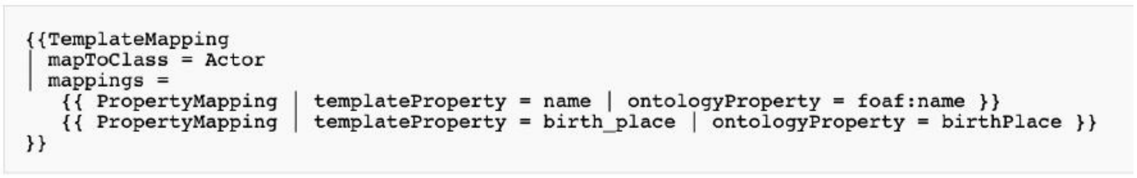
Later, in the extraction phase, this tree is forwarded to one of the DBpedia extractors. They consume the AST and return a set of RDF statements. The final step is the output, where RDF statements are written and become a DBpedia page.

There are 19 extractors that can be divided into four categories (LEHMANN, Jens et al., 2015):

- Mapping-Based Infobox Extraction: Manually written mappings relating Wikipedia to DBpedia ontology.
- Raw Infobox Extraction: Provides a direct mapping from Wikipedia's infoboxes to RDF.
- Feature Extraction: Uses extractors specialized in extracting a single feature from an article, for example, a label.
- Statistical Extraction: Aggregate data from all Wikipedia pages to find statistical measures of page links or word counts.

There are several infobox templates all over Wikipedia. They are defined by users and organized by categories, such as People, Singer and City. Each of these templates has properties that should belong to its resources (articles that belong to a specific template).

There are two possible ways a resource can inherit its template properties. The first is simpler, by substitution. In this method, the template content is copied to the resource only once, during its creation, and it never changes (figure 15).



```
{{TemplateMapping
| mapToClass = Actor
| mappings =
   {{ PropertyMapping | templateProperty = name | ontologyProperty = foaf:name }}
   {{ PropertyMapping | templateProperty = birth_place | ontologyProperty = birthPlace }}
}}
```

Figure 15 - Template mapping syntax. Source: http://mappings.dbpedia.org/index.php/How_to_edit_DBpedia_Mappings

Following, the second method is more complex, the transclusion. In this, the wiki text refers to the template, which means that every time the template changes, the properties inside the resources change too.

All the mappings supplied by the knowledge base are provided by the DBpedia user community, that creates and maps them from Wikipedia structures. They use the Resource Description Framework (RDF) as a flexible way to represent the extracted information and open it on the Web.

When it comes to DBpedia ontology, it is based on OWL and defines the structure for DBpedia. It describes its classes and properties and can be found in its mapping page[26].

According to Jean Nguema (2020), in a query he ran in 26/03/2020, there were 760 distinct classes inside DBpedia ontology, with 2727 distinct properties. Moreover, in the highest node of the ontology there is the class "owl: Thing", from which other classes inherit.

As the quantity of data on the Web increases, it is important to keep quality and consistency, which is a challenge to such a big dataset as DBpedia. Intending to help this repository to remain in its current position, as a central hub, this study analyzes its main issues, aiming to discover potential improvement points to be fixed, making it more complete and accurate for future studies and uses.

2.3 FAIR DATA PRINCIPLES

The European Open Science Cloud (EOSC)[27] was launched by the European Commission, the executive of the European Union that promotes its general interest, in 2016. However, it had the official launch event on November 23th, 2018, when they presented the EOSC portal and its governance structure.

[26] http://mappings.dbpedia.org
[27] https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-cloud

The EOSC was shaped in 2015 and was meant to both help Europe get a lead position in scientific data infrastructure and assure that European scientists were getting the full benefits of data-driven science, supporting open science and open innovation.

Open science is one of the goals Europe has set for its research and innovation policies. It refers to sharing all knowledge available using digital and collaborative technology. Its main focus is to change the way scientific publications occur nowadays, so instead of posting the result only at the end of the research, this would be done gradually throughout the entire process.

Another important objective is open innovation, focused on allowing open knowledge to circulate freely. Instead of focusing on the scientific process, this refers to opening up the innovation process to people with experience in other fields than academia and science, letting it be used in the development of new products and services.

For the EU to adopt defense and foreign policies, all Member States must agree unanimously. Aiming an ideal that could unite them, GO FAIR arises as a practical implementation of the EOSC[28] (COLLINS, Sandra et al., 2018), easily aligning their policies and investments in the area. Therefore, a new concept arises, the FAIR Data.

Despite all new technologies that have been created in recent years, there is still a lack of specifications, because, according to the seminal Royal Society report of 2012[29] (BOULTON, Geoffrey et al., 2012), research data being open is not sufficient. They need to be accessible, assessable, interoperable and usable. This way, artificial intelligence will be capable of identifying patterns and correlating data, leading them to be discovered, understood and used by other people.

FAIR relies on Findable, Accessible, Interoperable, and Reusable, the four main principles for scientific data management published in *Scientific Data* in 2016. As it was mentioned by Mark D. Wilkinson in the same article:

> There is an urgent need to improve the infrastructure supporting the reuse of scholarly data. A diverse set of stakeholders—representing academia, industry, funding agencies, and scholarly publishers—have come together to design and jointly endorse a concise and measurable set of principles that we refer to as the FAIR Data Principles. The intent is that these may act as a guideline for those wishing to enhance the reusability of their data holdings. Distinct from peer initiatives that focus on the human scholar, the FAIR Principles put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals. (WILKINSON, Mark D. et al, 2016, p.1)

[28] https://www.go-fair.org/2017/02/25/go-fair-european-open-science-cloud/

[29] https://royalsociety.org/~/media/Royal_Society_Content/policy/projects/sape/2012-06-20-SAOE.pdf

FAIR data is not meant to be open to anyone, it means that it is accessible to appropriate people, at an appropriate time and way (HODSON, S., et al, 2018). It depends on how this data are being used and their purpose, as if they are experimental or governmental.

Although FAIR principles[30] are not only for open data, it is important to understand their concepts in the context of EOSC and global drive towards science, so that a comparison between how open data in Brazil are and how they should be is inevitable.

The first pillar, Findable, means that data can be easily found, for both humans and computers. As the goal in the future is to cross different datasets, a way to find them automatically is crucial, and an essential part of the FAIRification process[31] (turning data into FAIR data). Furthermore, some steps are necessary to achieve this, being one of them, that data and metadata need to be assigned globally by unique and persistent identifiers.

Another principle is Accessible, which means that even though data can be found, they need to have appropriate authorization and a well-defined and universally implemented protocol. Accessibility in FAIR means that any human or machine should be able to access at least the metadata and use it across the internet.

Interoperable data means that it communicates with other similar data and, at the same time, interoperates with applications and workflows for storage, analysis, and processing.

Data need to be described using a vocabulary that contains the concepts they are representing, a community recognized specification. Technically, it means that the associated details are encoded using the same standard, that can be read by application systems.

Finally, there is the Reusable principle, that focuses on the reuse of data. It consists of well-described metadata that provide information about their provenance and steps to transform data into a more usable and understandable entity, allowing humans and machines to reuse it.

Although there are still many changes in the way, there is a more precise division of the GO FAIR structure that pays attention to every detail of the GO FAIR Implementation Network.

The Implementation Network (IN)[32] is a consortium dedicated to defining and creating materials and tools to reach the Internet of FAIR Data and Services (IFDS). It is self-governed and depends on people from all around the world to engage with it, ranging from individuals to whole organizations. (GO FAIR, 2017).

INs aim to implement the elements of a FAIR internet based on three pillars:

[30] https://www.go-fair.org/fair-principles/
[31] https://www.go-fair.org/fair-principles/fairification-process/

[32] https://www.go-fair.org/implementation-networks/

- GO Build: focused on creating FAIR technology that would allow a global infrastructure for seamless integration of data, tools and computing capacity
- GO Change: engaging around the cultural aspect of changing the current paradigm and establishing a new FAIR academic culture
- GO Train: aim to create a framework to rapidly train an abundance of competent, certified data scientists.

Finally, this all adds up to a common objective of disseminating FAIR concepts, in a way that more people can adhere to the culture and improve their data for a better global interconnection between them.

2.4 DATA PROFILING

As Garret Alley describes in his article written in 2019:

> Data profiling is a process of examining data from an existing source and summarizing information about that data. You profile data to determine the accuracy, completeness, and validity of your data. (ALLEY, Garrett, 2019)

Data profiling is important in projects and researches, as it is a usual task most people must have done in life, at least by eye-balling spreadsheets in order to find relevant information.

To start with, it incorporates several methods of data analysis and result generation, from finding the number of null values to the detection of approximate or conditional properties (NAUMANN, Felix, 2014).

There are some main use cases for data profiling, shown in table 2:

Table 2: Data profiling main use cases

USE CASE	DESCRIPTION
Query Optimization	Returns statistics about tables and columns (used to estimate the selectivity of operators)
Data cleansing	Used to prepare a data cleansing process (reveals inconsistent formatting, missing values or outliers)
Data integration	Used to explore the dataset before an integration (its size, data types, and semantics)
Scientific data management	Analyzes raw data to understand their structures and then devise an adequate schema
Data analytics	Analyzes data to help understand their current situations and appropriately configure tools

Source: Naumann (2014)

As in this research the main objective is to understand the current problems of diverse datasets, it was guided by data cleansing, in order to provide better data, to understand how the datasets are nowadays and how can they have some of their quality characteristics enhanced.

It is important to note that there are plenty of challenges when it comes to data profiling (NAUMANN, Felix, 2014). First, it can be useful to reveal which data are being used and which are not, leading to a better understanding of characteristics that may be causing it. Second, many existing profiling methods do not handle non-traditional data types, such as non-relational (e.g., LOD), non-structured (e.g., tweets), and heterogeneous (e.g., open government data), making room for the creation of new methods to support data profiling.

To sum up, data profiling is complex but extremely helpful when dealing with datasets with a huge amount of data, as DBpedia.

2.5 ETL4LOD

Combining heterogeneous data sources has always been a challenge, especially years ago, where there were no tools for dealing with this issue. In the 1980s, though, computer scientists started to investigate and develop software to make this possible.

The first one, driven by structured metadata, was designed by the University of Minnesota, in 1991. It was created for the Integrated Public Use Microdata Series (IPUMS), the world largest individual-level population database. It uses the so called ETL process, which means that it extracts, transforms, and loads data from different sources into one single store, making them compatible.

ETL stands for Extract, Transform and Load, the three functions that set an ETL workflow. Extraction means that the software will be doing data extraction from a source. Then, it will transform the data acquired following specified rules, altering their structure, and cleaning them. Finally, Load means that the software can deliver all data to another database, or even create a readable file for later use.

There are several ways to publish linked data. However, for this research, an ETL approach was the chosen alternative. The main reason is that ETL already comprises many necessary steps for publishing and managing LOD, such as the extraction of useful data from different sources, removal of data inconsistencies, and converting data from one format to another (CORDEIRO et al., 2011).

For the decision of an ETL strategy, ETL4LOD was a natural candidate, a framework from the LinkedDataBR project (CORDEIRO et al., 2011). This framework consists of a series of extensions built on Pentaho Data Integration (PDI)[34], as plugins to ETL and the data integration tool.

Besides the plugins originally built in the LinkedDataBR project, there were some other extensions developed afterwards, as ETL4LOD+ and ETL4DBpedia, aiming for the increase and improvement of the project functionalities.

At first, these were the primary plugins created by the GRECO group (Grupo de Engenharia do Conhecimento) in 2011 (CORDEIRO et al., 2011):

- Any23 Converter: converts data in formats accepted by LOD – N-triples, RDF/XML, and Turtle;
- Sparql Endpoint: allows SPARQL queries to run in specific endpoints;
- Sparql Update Insert: allows the manipulation of data inside an endpoint;
- Data Property Mapping: supports the data annotation using an ontology as reference;
- Object Property Mapping: supports the annotation of relations between data using an ontology as reference.

In 2018, ETL4LOD was upgraded to the last version of PDI, named as ETL4LOD+[35], on which the existing plugins were improved and others were created. Curcio (2018) created the following plugins:

- Owl Input: searches and selects ontology terms;
- Link Discovery Tool: supports linking data sources.

Finally, in 2020, Ngomo (2020) created a group of plugins that complemented DBpedia current data extraction from Wikipedia, the ETL4DBpedia[36]. His work focused on supporting the upload of data from public organizations or other collaborators to DBpedia in a semiautomatic way. The implemented plugins were:

- Domain Data Transformer: extracts, clean and transforms a set of domain data;
- Templates Maintainer: gets the template names that match the string inputted by the user;
- DBpedia Mappings Maintainer: returns a list of templates and a field indicating whether they were persisted or not;

[34] https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho-platform/pentaho-data-integration.html
[35] https://github.com/johncurcio/ETL4LODPlus
[36] https://github.com/JeanGabrielNguemaN/ETL4DBpedia

- Template Selector: gets the classes that are more connected to the concepts chosen by the user;
- Template Mapper: allows the user to choose a template and line up its properties, to be used in the creation of an infobox;
- Article Checker: checks if the potential article already exists in Wikipedia;
- Article Content Builder: generates the potential article content with data received from the Template Mapper step;
- Article Publisher: joins the data in a wiki text format and publishes them on Wikipedia.

Moreover, as the major focus of his research was to load data directly into DBpedia, his plugins were built aiming to support this objective, by helping the user in the necessary steps to build a new article. Nonetheless, it also has some plugins that either check whether there is a template that matches the prefix written by the user or that maps a domain field to a property in the infobox.

Therefore, despite the fact that there are some plugins in this project that deal with more primitive parts of DBpedia extraction process, such as templates, they do not actually assess them, as they focus on finding matches or their mappings.

Complementarily, ETL4Profiling gives a deeper look into templates and their properties, not only searching their mappings but also trying to find inconsistencies in DBpedia current data, giving a better prospective of the most common problems, that can later be used by projects as ETL4DBpedia, as explained minutely in chapter 5.

With all being said, ETL4LOD is extremely complete nowadays, what has led to its use as a basis for both this research and the plugins implementation.

2.6 RELATED WORD

As mentioned previously, ETL4DBpedia works in a complementary way to ETL4Profiling, as the first analyze the completeness of new data while the second, of the current data. However, this is not the only related work this project has.

In the DBpedia mapping page, there is a part containing the statistics for each DBpedia[37] edition and their templates[38]. Conversely, it was discovered that this page consists of

[37] http://mappings.dbpedia.org/server/statistics/pt/
[38] http://mappings.dbpedia.org/server/templatestatistics/pt/?template=Info/Munic%C3%ADpio_do_Brasil

percentages about how much of Wikipedia properties are mapped on DBpedia, instead of evaluating its present data.

In some kind of manner, the statistics provided by DBpedia and the ones that could be found by this work could complement each other, because while one analyzes the first transformation step, from Wikipedia to DBpedia, the other studies the next step, the completeness of these mapped properties that were inherited by the template resources.

3 ETL4PROFILING

Originally, the main purpose of this work was focused on FAIR principles, finding a way to improve DBpedia data quality to retrieve trustworthy and correct data that could be open and reused by researches and other projects.

3.1 FAIR ANALYSIS

As the study began, problems also started to appear. Among FAIR principles, one that stood out was the Reuse (R1). It focuses not only on the potential data has to be found, but also on the ability the user has to decide if they are useful in a particular context. Moreover, it also describes that metadata authors should be as generous as possible when providing metadata, because it is easier to find and reuse data when there are plenty of labels attached to them.

When analyzing the retrieved data, FAIR discusses openness, sharing and reuse of data. Nevertheless, assessing them through another perspective, the data quality, can also provide startling information to assist them in FAIR specifications, by informing major attention spots that can be fixed, in order to provide fuller metadata and, as a consequence, more easily found and reused data.

3.2 DBPEDIA ANALYSIS

DBpedia has a great quantity of problems with its data as previously introduced, such as wrong classification of entities and missing properties. However, the biggest obstacle is not only ignorance of the main flaws, but not having their real extent.

The resource Baldim, for instance, a Brazilian county of Minas Gerais, possesses a huge amount of properties that belong to its template, "Info/Município do Brasil". Conversely, it also has some missing properties, such as “Padroeiro” and "região_metropolitana". Furthermore, some of the resource properties are not even mapped on the template.

Although it is effortless to compare the properties from one resource to the template, in a broader spectrum it turns out arduous, given the fact that a single template can have more than a thousand associated resources.

In addition, it is widely known that DBpedia has several issues, but due to its huge number of templates, it is a challenge to identify its true extension, since manually evaluating every and each resource inside a template is virtually impossible.

Consequently, it encouraged the implementation of ETL4Profiling, a functionality that automated the investigative process and brought to light statistics about templates, as well as resources and properties, allowing a better recognition of the issues and the required work to fix them.

3.3 ETL4PROFILING FIRST STEPS

The first step in the development was to discover what was already available on the Web and what was missing.

As described on the subchapter 2.6, there are already two projects related to DBpedia data quality, which are its own mapping statistics and ETL4DBpedia. While the first focusses on generating data about how much of Wikipedia is mapped on DBpedia, the second supports the insertion of new complete and correct data into the dataset, aiming to increase the repository data quality.

Meanwhile, it was sought, but not found, a framework that performed an analysis in its current data regarding their completeness, instead of mapping. Therefore, taking into account the importance of assessing what is already there, an evaluation of DBpedia existent data was essential.

After discovering the right data to evaluate, it was time to decide which kind of framework should be developed.

The initial idea was to make a website that would retrieve all DBpedia templates with their respective resources and completeness, ranked from the most complete to the least. However, with this approach all the information would be locked in a single place, contrary to the intended openness principle. Soon this was discarded another possibility was raised, to extend ETL4LOD plugins to analyze datasets.

As before, the idea of creating Pentaho plugins suited the proposal of the project. New data are being created every day, and it is indispensable to make them as complete and correct as possible, especially if they are going to be stored in such a dominant repository as DBpedia.

Considering this, the logical choice was to implement the plugins following the ETL (Extract, Transform, Load) workflow, mostly ETL4LOD, because of its positive ratings in previous uses.

Whereas ETL4LOD contains plugins that run SPARQL queries in certain endpoints, convert data, search ontology terms and even support data annotation, none of them are focused on data profiling, especially on DBpedia.

Furthermore, ETL4Profiling is meant to extend ETL4LOD, contributing with profiling tools to help in the analysis of DBpedia and other datasets. After this definition, it was necessary to decide which statistics would be calculated.

Starting from a specific part and then generalizing it is easier than the opposite, therefore the plan was to create DBpedia plugins at first glance and later generalize them by creating steps that would work with any dataset.

Because the major problems found previously, during a Scientific Initiation research, occurred with missing predicates/properties it was fair to start with them. Moreover, DBpedia properties come from a defined source, the template that specifies them.

DBpedia has a mapping page that has all its templates and their properties, the ones their resources should have, setting it as the starting point.

3.4 TOOLS USED IN THE IMPLEMENTATION

For this implementation, some external tools were used. They were meant to support the creation of plugins and facilitate their development.

The preferred IDE used was Eclipse IDE for Enterprise Java Developers in version 2019-12 (4.14.0), because of the facilities brought when working with Maven for dependencies management. The equipment used was a Macbook Pro 2019, with macOS Catalina (Version 10.15.2), Intel Core i5 Quad-Core (2,4GHz) processor and a RAM memory of 8GB.

3.4.1 JSoup[39]

Jsoup API is a Java library that works by fetching a chosen URL and delivering the extracted data, making it possible to also manipulate them, giving the user the opportunity to take a deeper look in the page elements. It uses HTML5 DOM methods and CSS selectors, also implementing the WHATWG HTML5 specifications.

The library usage is extremely simple, as a documentation is available in its website with the correct directions to its usage in a project.

First, it works with a huge variety of functionalities. It has methods to parse an HTML from a string, a file, or even connect to a URL. It also provides functions to extract elements

[39] https://jsoup.org/

from the parsed HTML, with selectors or DOM methods. Moreover, it allows their manipulation, by setting attributes or indeed an HTML.

With the main goal of finding DBpedia templates and resources properties, JSoup API was used parsing DBpedia mapping page and the resources URIs, getting the necessary information while the plugins were still running.

3.4.2 Apache Maven[40]

Apache Maven is a Java software for project management based on the project object model (POM). It is a built automation tool that describes the project with its dependencies, external components, compilation order, directories and required plugins.

To make it simpler for the user, Maven downloads Java libraries and its plugins dynamically from one of their repositories, storing them in a local cache, as they are packaged in jar files (Java Archive).

As the plugins required some external libraries and seeking their easier usage, Maven was chosen as the dependency manager, with the responsibility of compiling and installing the components inside Kettle.

3.4.3 Apache Jena[41]

Apache Jena is a Semantic Web framework built for Java. It provides a complete environment with tools to run SPARQL queries in datasets, to deal with RDF, and even an API for extracting data and writing it to RDF graphs.

In this study ARQ was the most used tool, a SPARQL processor for Jena. It was used to run queries and get the resources from a template. With Jena, it was possible to set the URL where it should run (DBpedia SPARQL endpoint), the query, its parameters and receive the expected results, making the search for data much simpler.

3.5 ETL4PROFILING PLUGINS

Plugin can be defined as a program that is written to be added to a bigger program (STERNE, Jonathan, 2019), also known as extension module. They serve to add new

[40] https://maven.apache.org/what-is-maven.html
[41] https://jena.apache.org/

functionalities to the program that hosts them and provide, in general, a special or highly specific feature. Usually, they tend to be small and light, only used on-demand.

In the scenario of this study, the plugins were created with the sole goal of calculating statistics about the templates, resources and properties, related to their completeness, being added to improve ETL tools.

The main attention points were about subject predicates. When it comes to DBpedia, they were its templates, their resources, and properties. The plugins did not focus only on its Portuguese version but had a wider view providing analysis about other available versions as well, making a better comparison possible.

The plugins can be grouped by their functions (table 4) or ontologies (table 3) they analyze.

Table 3: Grouping plugins by ontologies they analyze

ONTOLOGY	PLUGINS
Template (DBpedia)	Template Property Analyzer, Template Resource Input Analyzer, Template Resource Analyzer
Resource (DBpedia)	Resource Properties Analyzer, Resource Input Analyzer, DBpedia Triplification (triplifies resources), Get DBpedia Data (retrieves resources)
Property (DBpedia)	Property Analyzer, Get DBpedia Data (retrieves properties)
Subject (any dataset)	Inner Profiling, Merge Profiling

Source: The author

Table 4: Grouping plugins by functions

FUNCTION	PLUGINS
Analyzes DBpedia completeness	Template Property Analyzer, Template Resource Analyzer, Resource Properties Analyzer, Property Analyzer
Analyzes input completeness	Template Resource Input Analyzer, Resource Input Analyzer
Analyzes dataset completeness	Merge Profiling, Inner Profiling
Triplifies DBpedia	DBpedia Triplification
Gets resources and properties from DBpedia	Get DBpedia data

Source: The author

The five functions the plugins can be grouped by were chosen because of their relevance.

As DBpedia is the main case study for this research and is based on a specific ontology, it was necessary to build plugins specially for its analysis. Moreover, it is relevant to check the completeness of a resource and realize how it affects the entire template percentage, for instance, how the missing properties in "Tocantins" may affect the percentage of "Info/Município do Brasil".

On the other hand, it is also important to compare different sources to DBpedia data, and even more, to any other dataset. For example, it is both possible to check if all Brazilian counties are inside DBpedia, and if the data in a newer version of "Dirty List of Slave Labor" have some missing predicates.

For this reason, some plugins need an input to compare data, as Template Resource Input Analyzer, while others analyze the chosen option automatically, already having all the information needed (figure 16).

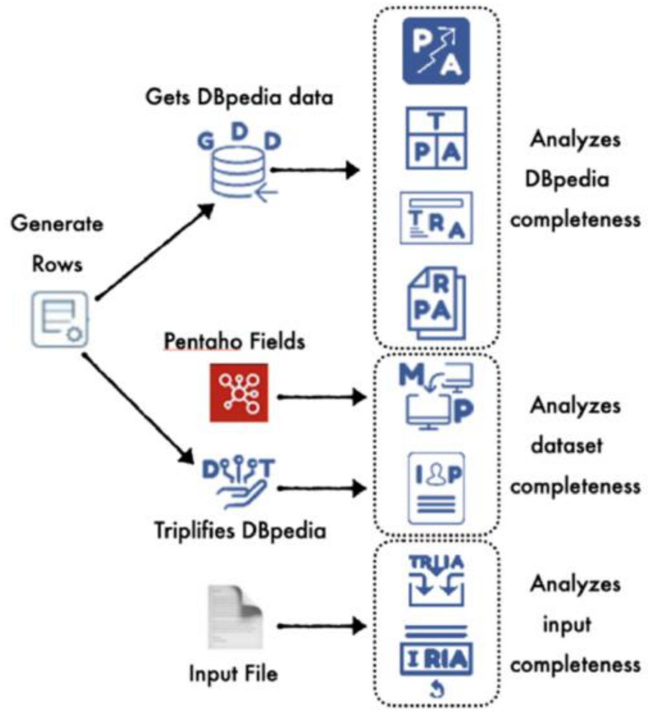

Gets DBpedia data
Generate Rows
Pentaho Fields
Triplifies DBpedia
Input File
Analyzes DBpedia completeness
Analyzes dataset completeness
Analyzes input completeness

Figure 16 - Possible interconnections between Plugins

Another relevant observation is the opportunity of using DBpedia data in a different context, for future analysis, being that the motivation behind the creation of Get DBpedia Data and DBpedia Triplification, granting the user the possibility to use them even with an external plugin.

With that mentioned, it is now time to closely understand each plugin.

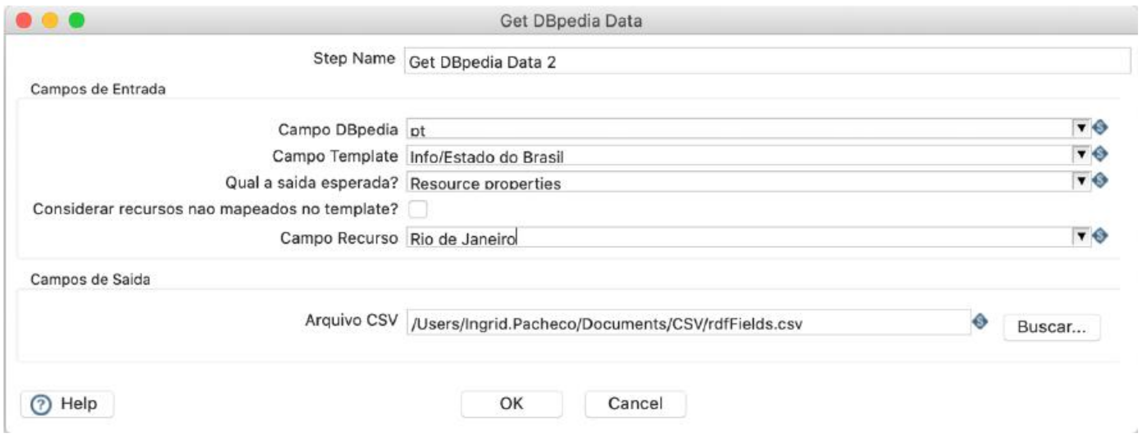
3.5.1 Get DBpedia Data

Get DBpedia Data (figure 17) is certainly one of the most powerful plugins of the project. It was created aiming the usage of DBpedia data in contexts other than only the profiling meant for this study.

For the sake of this research, it was used mainly for extracting and managing DBpedia data, to find the profiling of its structure. However, it can also be used in different ways, for instance, as the input for another plugin or even in transformations outside PDI.

The plugin receives the DBpedia version, the template and the resource field, if the user decides to get the resource data. Moreover, it was designed to deliver four kinds of information:

a) Template properties: delivers the template properties, with DBpedia version, template and property as fields;
b) Template resources: delivers the template resources (names), with DBpedia version, template and resource as fields;
c) Template resources properties: delivers all template resources with their respective properties. It has the DBpedia version, template, resource, property name, property value and property type as fields;
d) Resource properties: delivers an individual template resource with its properties. It has the DBpedia version, template, resource, property, property value and property type as fields.


Get DBpedia Data

Step Name: Get DBpedia Data 2

Campos de Entrada

Campo DBpedia: pt
Campo Template: Info/Estado do Brasil
Qual a saida esperada?: Resource properties
Considerar recursos nao mapeados no template? ☐
Campo Recurso: Rio de Janeiro

Campos de Saida

Arquivo CSV: /Users/Ingrid.Pacheco/Documents/CSV/rdfFields.csv

Figure 17 - Get DBpedia Data plugin

When the user chooses the version they want to study, a Web scraper is done in the DBpedia mapping page[42] (Appendix A), loading all template choices of the selected DBpedia. As soon as the options are loaded, they appear in the Template field and the user can decide which template they want to analyze.

[42] http://mappings.dbpedia.org/index.php/Main_Page

Following, if the information chosen is Template resources, the program tries to run a SPARQL query getting all the resources from the template (figure 18).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: ?dbpUrl
SELECT DISTINCT ?uri ?name WHERE {
   ?uri dbp:wikiPageUsesTemplate ?templateUrl.
   ?uri rdfs:label ?label.filter langMatches(lang(?label), ?language).
   BIND(STR(?label)AS ?name).
}
```

Figure 18 - SPARQL query to get resources from template

Meanwhile, for the Template properties, it fetches the DBpedia mapping page to get the template properties.

In addition, for the Template resources properties, besides getting the resources and template properties, it runs another Web scrapping, this time hitting the URI of each resource found and getting their properties with values and types (Appendix B).

Finally, for the Resource properties, it does the same processing mentioned before, but only for a specific resource.

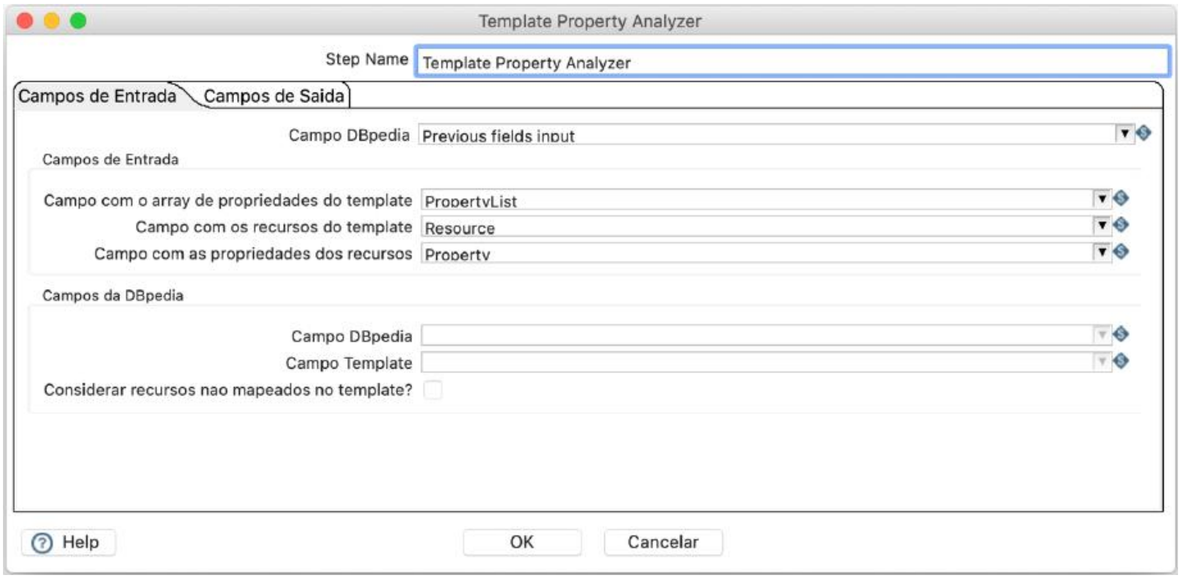
It is extremely tiring the fact that these information are not in the same place and so many processes have to be executed to get them, but it was a necessary procedure, to perform the most complete analysis.

After all data are discovered, the plugin writes them in a CSV file and outputs them in the PDI field. This usage was thought in a way of complementing the inputs for other plugins in this project.

3.5.2 Template Property Analyzer

Template Property Analyzer verifies if the template properties are being used by their resources, bringing the completeness of each property and, consequently, the template completeness. With this it is possible to identify which properties are least used by the resources, leading to an easier identification of the points to take action upon, fixing them and making the resources more complete.

The plugin has two possible options for input. The first is by getting the previous plugin output (figure 19), receiving the fields where the template properties, resources and the resource properties are. The second is by DBpedia fields, where the user selects the template they want to evaluate.



Template Property Analyzer

Step Name: Template Property Analyzer

Campos de Entrada | Campos de Saida

Campo DBpedia: Previous fields input

Campos de Entrada

Campo com o array de propriedades do template: PropertyList
Campo com os recursos do template: Resource
Campo com as propriedades dos recursos: Property

Campos da DBpedia

Campo DBpedia:
Campo Template:
Considerar recursos nao mapeados no template?

Figure 19 - Template Property Analyzer receiving previous step's outputs

Finally, with all the essential data, it analyzes them and gets the statistics.

As the goal of this plugin is to get the percentage of the template, based on its properties, the program gets the template properties and compares them to each resource property, finding the exact number of times a property appeared.

It does the calculation by comparing the resources they were in it to the number of times they should have appeared, following the expression: 100 * (resources the property was in) / quantity of resources.

The results found are sorted according to the order chosen and saved in a CSV file, containing the properties, the number of times they appeared in resources, the times they should have appeared and the completeness. Furthermore, a short report containing slightly more verbalized information is also written to a TXT file, besides the output in the PDI field.

3.5.3 Template Resource Analyzer

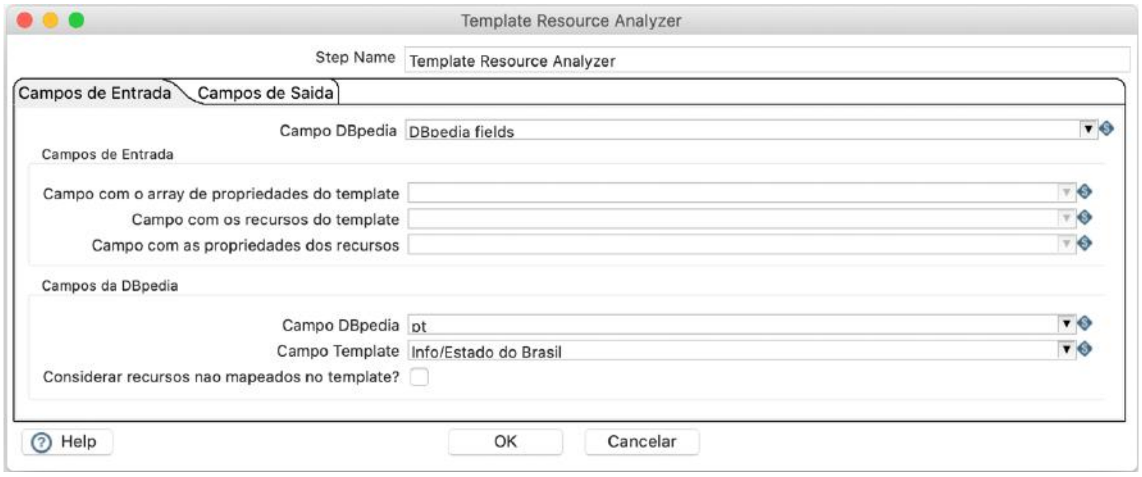
Template Resource Analyzer verifies the completeness of every template resource using their properties as the comparison tool.

The plugin describes how many template properties belong to each resource as well as the resource properties that are not mapped on the template and, accordingly, do not belong to it.

Observing that a template can have numerous resources (more than 1000), the plugin works to identify which of them are more incomplete and need to receive attention faster.

Besides, it is possible to identify how much the template can become more complete by the number of not mapped properties on it.

As well as the last plugin, it receives either the DBpedia fields (figure 20) the user wants to evaluate or the fields from the previous step.


Template Resource Analyzer

Step Name: Template Resource Analyzer

Campos de Entrada | Campos de Saida

Campo DBpedia: DBpedia fields

Campos de Entrada

Campo com o array de propriedades do template:

Campo com os recursos do template:

Campo com as propriedades dos recursos:

Campos da DBpedia

Campo DBpedia: pt

Campo Template: Info/Estado do Brasil

Considerar recursos nao mapeados no template? ☐

Figure 20 - Template Resource Analyzer receiving DBpedia fields

It gets both the template properties and the template resources, comparing the first to the properties found on each resource. As the intention is to find the resources completeness, it first finds their missing properties (template properties that are not in the resource) and not mapped properties (resource properties that are not in the template), and then calculates the completeness based on the expression: 100 * (resources properties – not mapped properties) / (template properties).

Lastly, it sorts the resources using their percentages.

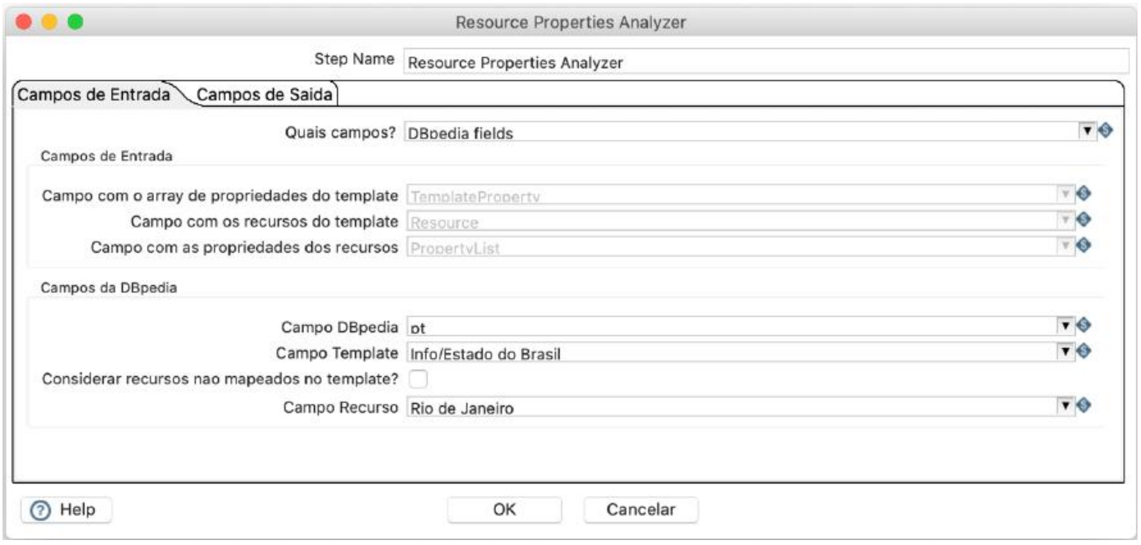
The results are then saved in a CSV and in a TXT files, containing the resource, the number of template properties found on it, the properties that are missing, the ones that are not mapped and their completeness, besides the output in the PDI field.

3.5.4 Resource Properties Analyzer

Resource Properties Analyzer aims to find the completeness of an individual resource, using its properties and the template properties to accomplish it, comparing them, and indicating which are missing or not mapped.

This is meant to identify if the template properties are inside the resource and if the resource properties are mapped on the template and thus if they are not exclusivities of the resource.

The plugin receives the usual inputs, but also the specific resource the user would like to evaluate (figure 21).


Resource Properties Analyzer

Step Name: Resource Properties Analyzer

Campos de Entrada | Campos de Saida

Quais campos? DBpedia fields

Campos de Entrada

Campo com o array de propriedades do template: TemplateProperty

Campo com os recursos do template: Resource

Campo com as propriedades dos recursos: PropertyList

Campos da DBpedia

Campo DBpedia: pt

Campo Template: Info/Estado do Brasil

Considerar recursos nao mapeados no template?

Campo Recurso: Rio de Janeiro

Figure 21 - Resource Properties Analyzer receiving DBpedia fields

In this plugin, another ontology analysis is done, this time not focusing on the template itself, but in one of its resources.

With all the inputs selected, the program gets the resource properties and compares them to the template properties, returning which are missing in the resource, not mapped on the template, or even in both of them.

To calculate the completeness, it compares the two lists and makes the following calculation: 100 * (resource properties – not mapped properties) / (template properties).

It provides the user the option of choosing which properties they want, with four possibilities: the ones that are on both the template and resource, only on the resource, only on the template or all properties.

The result is saved in a CSV and TXT files, besides the PDI field, with the properties, if they are mapped on the template, if they are on the resource, and finally the completeness of the resource, based on the properties it has and the quantity it should.

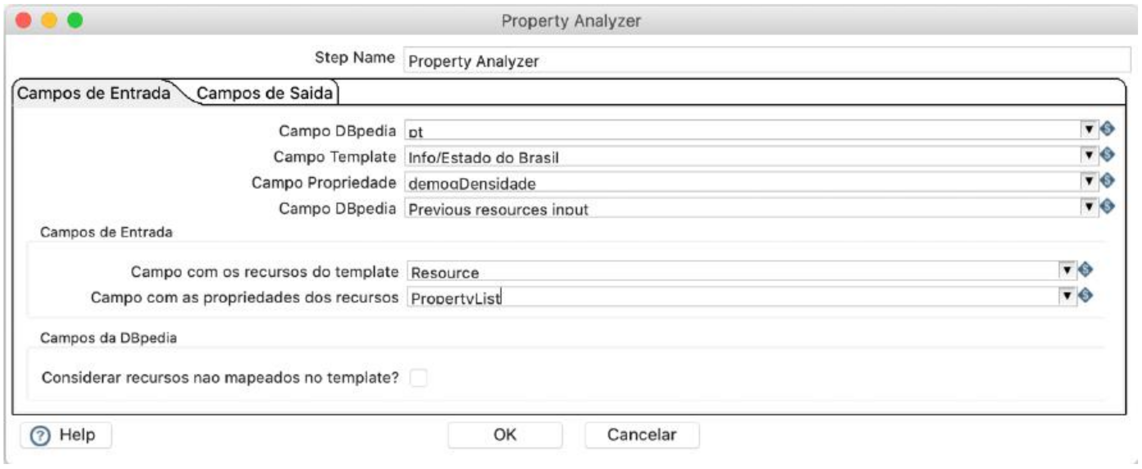
3.5.5 Property Analyzer

Property Analyzer studies another relevant peace of DBpedia, the properties.

It is going to study the presence of the property in every template resource, showing the user, depending on their choice, the resources that have or do not have that property. Besides, it also returns its completeness based on how many resources it is present compared to the amount of template resources.

To calculate the completeness, it makes the following calculation: 100 * (resources the property is inside) / (template resources).

This plugin receives, besides the usual inputs, the property chosen from DBpedia (figure 22).


Property Analyzer

Step Name: Property Analyzer

Campos de Entrada | Campos de Saida

Campo DBpedia: pt
Campo Template: Info/Estado do Brasil
Campo Propriedade: demoqDensidade
Campo DBpedia: Previous resources input

Campos de Entrada

Campo com os recursos do template: Resource
Campo com as propriedades dos recursos: PropertyList

Campos da DBpedia

Considerar recursos nao mapeados no template? [ ]

Figure 22 - Property Analyzer receiving DBpedia fields

Differently from the other plugins, it gives a deeper view inside the properties, instead of only using them as an evaluation parameter.

The main analysis it provides is the completeness of the property, but in a fuller view, taking into consideration that it is possible to find exactly the resources that are lacking this property, it also helps to encounter the resources that could evolve by receiving a value in it.

As a result, the plugin returns the option chosen by the user, retrieving the names of the resources, and whether they have the property or not. Lastly, with the exact number of times the property was found and the quantity of resources, it returns its completeness, that is saved in the PDI field, and in a CSV and TXT files.

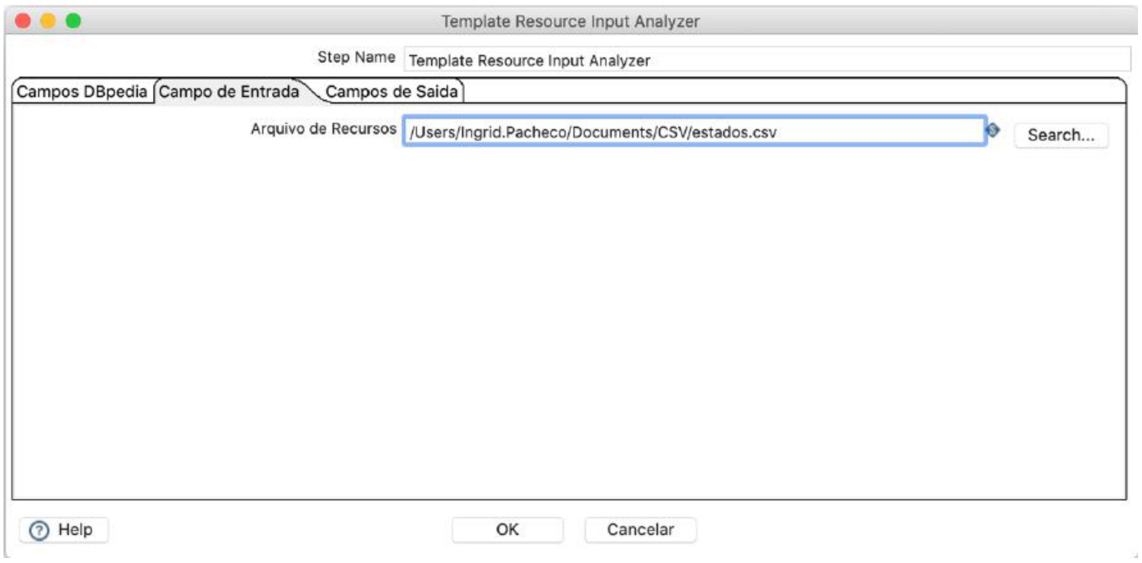
3.5.6 Template Resource Input Analyzer

Template Resource Input Analyzer aims a different goal than the other plugins previously presented. It is going to compare the resources that it receives with the resources found on a DBpedia template.

It is significant because it allows the user to do a fuller analysis on the resources the template should have, for instance, the counties that should be inside "Info/Município do

Brasil". It returns, depending on the user's choice, the resources that are available on both (the CSV and DBpedia), or only in one of them.

The plugin receives the DBpedia version it has to evaluate, the template and the CSV file containing the resources that should belong to it (figure 23).



Step Name: Template Resource Input Analyzer

Arquivo de Recursos: /Users/Ingrid.Pacheco/Documents/CSV/estados.csv

Figure 23 - Template Resource Input Analyzer receiving the input file with Brazilian states

This plugin is different, so as its function. The ones presented before had the function of analyzing DBpedia data and find some completeness. Although this one also delivers the percentage, it was created for the sake of comparison.

The user inputs a file containing the resources that should be inside the selected template. This time, the reference comes from an outside source, instead of being totally calculated based on DBpedia.

It compares all the resources from the file and returns both the ones that are not and the ones that are on DBpedia. Moreover, it gives the completeness based on the data the file consists, following the expression: 100 * (resources on DBpedia – resources that are not on the file) / resources on the file.

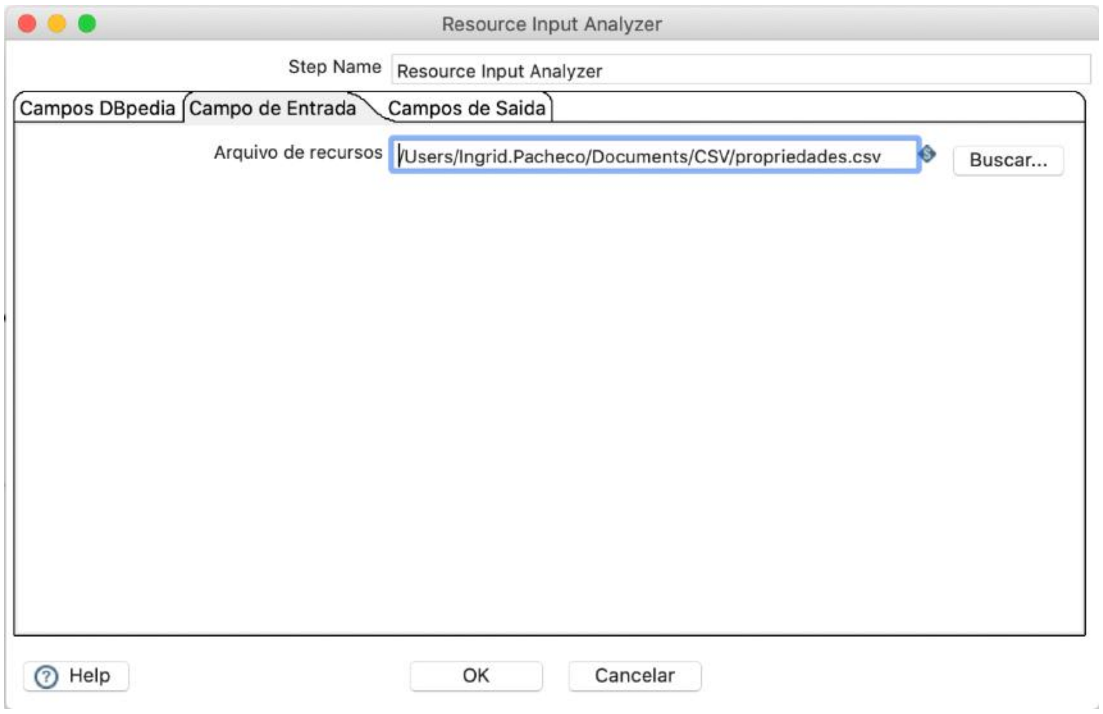
This way, it is possible to check if a template has all the resources it should have, and in case there are any missing, enables their possible creation in the future.

3.5.7 Resource Input Analyzer

Resource Input Analyzer compares the properties received as input with the ones on the chosen template, showing which are missing, bringing to light the input's actual level of completeness.

The plugin is meant to simulate the input of a new resource in the selected template, elucidating which properties would be missing considering the template properties.

It receives as input the DBpedia version it has to evaluate, the template, and the input file containing the properties for comparison (figure 24).


Resource Input Analyzer

Step Name: Resource Input Analyzer

Campos DBpedia | Campo de Entrada | Campos de Saida

Arquivo de recursos: /Users/Ingrid.Pacheco/Documents/CSV/propriedades.csv

Figure 24 - Resource Input Analyzer receiving input file

As a matter of fact, it is one of the most fascinating plugins as it has the function to compare and give the completeness of a resource considering possible properties, to check how complete it is.

If plenty of resources are lacking properties and are incomplete, there should be a willingness to fix them and make them more correct, for the sake of open data. As it is not possible yet to change all the oldest data, an initiative to assist in the creation of new data would be helpful, as is meant to happen with Resource Input Analyzer.

In regard to how the plugin works, it receives the selected template and the properties. Later, it gets the templates properties and one by one checks whether they are mapped or not on the input, calculating the completeness using: 100 * (properties on the file – properties not mapped on the template) / (template properties) .

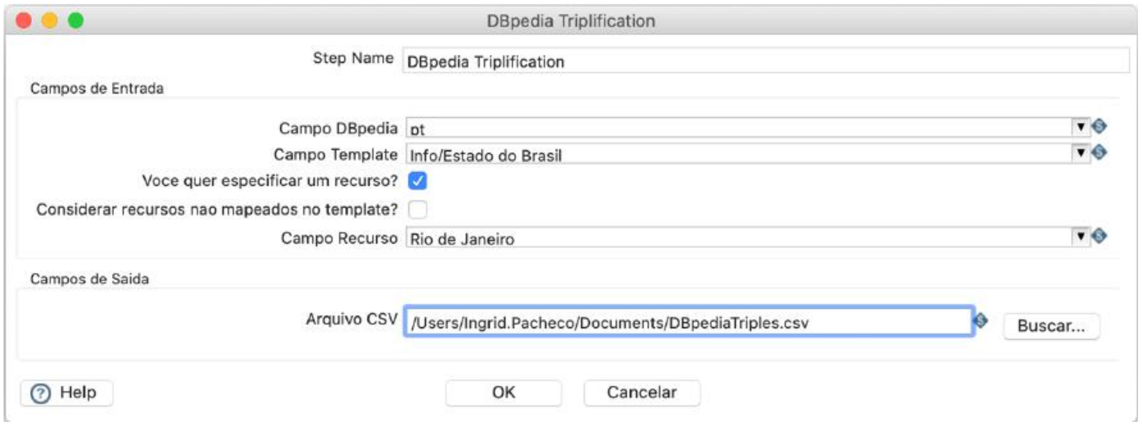
After getting all the required information by checking the properties, it knows which of them are not inside the input and returns the properties that are missing.

With this, there is a big possibility the new data will be more complete and correct when stored on DBpedia, increasing the completeness of the dataset.

3.5.8 DBpedia Triplification

After the conclusion that a deeper profiling on DBpedia was possible, the project got vaster and a generic profiling was sought. Moreover, aiming the usage of DBpedia resources with other datasets, a triplification of its data was fundamental.

DBpedia Triplification (figure 25) triplifies either all the resources from a template or an individual one, depending on the user's choice. It was developed focusing on using DBpedia resources as an input for future comparisons.



DBpedia Triplification

Step Name	DBpedia Triplification

Campos de Entrada

Campo DBpedia	pt
Campo Template	Info/Estado do Brasil
Voce quer especificar um recurso?	✓
Considerar recursos nao mapeados no template?	
Campo Recurso	Rio de Janeiro

Campos de Saida

Arquivo CSV	/Users/Ingrid.Pacheco/Documents/DBpediaTriples.csv

Figure 25 - DBpedia Triplification receiving DBpedia fields

It receives the DBpedia version the user wants to assess, with template and resource. Then, the plugin gets the DBpedia data and processes them in one of the following paths:

a) If a resource was chosen it gets all its properties along with their values and create a triplification using the resource URI, the property and the value.
b) If a template was chosen, the plugin triplifies all of its resources. First, it gets the resources, and for each of them does the proceeding mentioned above, getting their properties and values and creating a triplification for them.

Moreover, in both cases, it has to provide the correct type for the values, and therefore, it analyzes if they are a literal, number or another resource, to complement with proper information. The treatment is the following for each of these types:

a) Literal: If the value is a literal it is added between quotation marks and followed by the DBpedia version. For example, "Minas Gerais" would become "Minas Gerais"@pt .
b) Number: Being a number, the modification would be the insertion of the value's type at the end of the line, whether it is an integer or double. For instance, the double number 4.4 would turn into "4.4"^^<http://www.w3.org/2001/XMLSchema#double> .
c) Object: The last possibility is when a value is an object, and its URI has to be addressed in the transformation. As the value has the information that it is a resource, the only

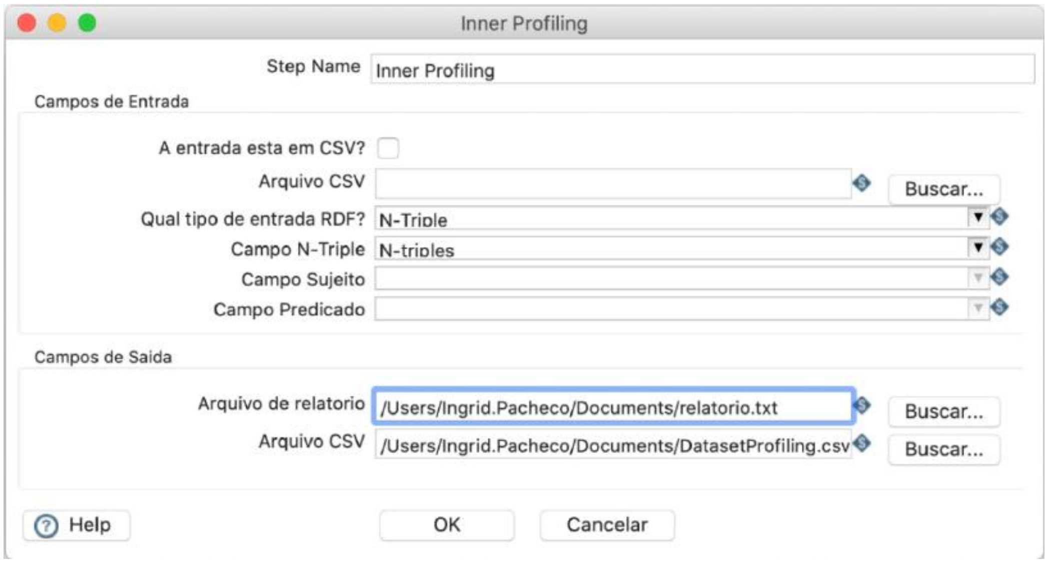
modification is the addition of "http://pt.dbpedia.org" in the begging of it. For example, if the value is another resource named "São_Paulo", it would have to be converted into "<http://pt.dbpedia.org/resource/São_Paulo>".

Lastly, the N-triples created are saved in a CSV and TXT files, besides the PDI field.

3.5.9 Inner Profiling

As the first plugin focused on doing generic profiling of diverse datasets, the Inner Profiling (figure 26) has the objective of, as the name describes, doing a more profound investigation inside the dataset given.

Essentially, the plugin will receive the dataset in one out of the two available formats, either in N-Triples or subject/predicate fields. Additionally, it also gives the user the opportunity to upload their dataset in a CSV file, granting more flexibility in the use of the tool.


Inner Profiling

Step Name: Inner Profiling

Campos de Entrada

A entrada esta em CSV? ☐
Arquivo CSV:
Qual tipo de entrada RDF? N-Triple
Campo N-Triple: N-triples
Campo Sujeito:
Campo Predicado:

Campos de Saida

Arquivo de relatorio: /Users/Ingrid.Pacheco/Documents/relatorio.txt
Arquivo CSV: /Users/Ingrid.Pacheco/Documents/DatasetProfiling.csv

Figure 26 - Inner Profiling receiving previous step's fields

The comparison done is based on the predicates each subject has, comparing them to an array with all the predicates found in the dataset.

To calculate the completeness, it follows the expression: 100 * (subject predicates) / (total of predicates).

Moreover, the array formed by all the predicates is constructed partially with every row's information. If a new predicate appears, independently of the subject, and it is not inside the array yet, it is included, aiming to become the main source of comparability to the subjects, allowing the discovery of their completeness.

As the processes for this plugin are more complex than the others, a deeper explanation is necessary.

The steps for the complete comparison are:

a) The predicate in the row is added in the array of predicates of the subject.
b) Later, it does an assessment to validate if the actual predicate was in the list of missing predicates of the subject. This may happen because while the rows are being processed, the list of missing predicates for each subject is being updated with each row's information. If the predicate is inside the list of missing predicates it is removed.
c) Then, it checks the array containing all the predicates found till the moment and verifies the ones that are not in the array of predicates of the subject, which means that they are missing. If it happens, they are added in the missing predicate list of the subject.
d) The predicate is added in the HashSet incorporating all the predicates.
e) Lastly, another round of comparison is done, but this time to check missing predicates for the other subjects of the input. Sometimes a missing predicate can appear after all the occurrences of one resource are done, and in some manner, this predicate has to be added in the list of missing predicates of the subject, giving reason for a second round of evaluations.

Finally, the results are delivered in a CSV and TXT file, besides the PDI field, pointing out the missing predicates for each subject.

3.5.10 Merge Profiling

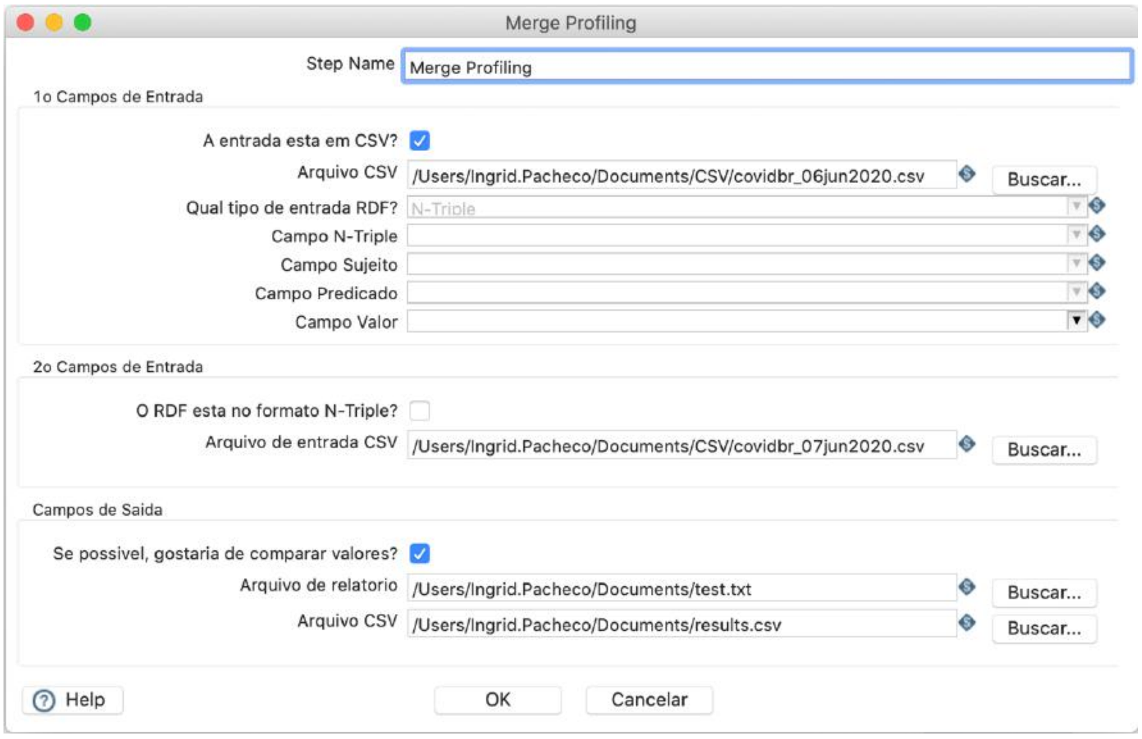
The final plugin is also one of the most fascinating, because of its value for data research.

Merge Profiling (figure 27) intends to compare distinct datasets, clarifying their missing predicates or different values and indicating what each subject has lost and gained correlated to another version.

The first target was to enable the study of the same dataset in differing renditions, explaining the modifications they suffered.

The plugin was then accommodated to be used also in situations where it is desirable to compare distinctive datasets with the same subjects, for instance, different COVID-19 datasets identified by a county, to check the missing information that could be integrated.

Primarily, it obtains the first and second datasets, which can either be in CSV or fields achieved by past plugins. Moreover, when it comes to the second dataset, it is also necessary to specify if it is triplified (N-triples) or in subject/predicate/value format.



Merge Profiling

Step Name: Merge Profiling

1o Campos de Entrada

A entrada esta em CSV? ✓
Arquivo CSV: /Users/Ingrid.Pacheco/Documents/CSV/covidbr_06jun2020.csv
Qual tipo de entrada RDF? N-Triple
Campo N-Triple:
Campo Sujeito:
Campo Predicado:
Campo Valor:

2o Campos de Entrada

O RDF esta no formato N-Triple? ☐
Arquivo de entrada CSV: /Users/Ingrid.Pacheco/Documents/CSV/covidbr_07jun2020.csv

Campos de Saida

Se possivel, gostaria de comparar valores? ✓
Arquivo de relatorio: /Users/Ingrid.Pacheco/Documents/test.txt
Arquivo CSV: /Users/Ingrid.Pacheco/Documents/results.csv

Figure 27 - Merge Profiling receiving datasets

Regarding its execution, it goes through every subject inside the datasets creating a dictionary, once again, correlating each subject with their predicates.

First, it saves all the predicates from the second dataset in an array related to the subjects they are in. Then, is does the same procedure for the first dataset, storing the subjects names in an array for future comparison.

The comparison is later done by checking which predicates from the first dataset are missing in each subject of the second, and the opposite situation too.

The calculation involves the following expression: 100 * (predicates from the first dataset – predicates that are not in the second dataset) / (predicates from the second dataset).

Moreover, another step it can provide, if the user decides to, is the value’s comparison. For each subject that has the same predicate in both datasets the values can be compared, considering they are numbers, and the plugin returns the main difference between them with the percentage of loss or gain, considering the expression: 100 * (value from the second dataset) / (value from the first dataset).

Conclusively, it returns in both a CSV and TXT file the results found, besides the PDI field, making it possible to discover if the same dataset lost any information in a newer version or even if a subject is more incomplete than another from the same source.

4 APPLICATION EXAMPLES

To test the ETL4Profiling plugins presented above, two main use cases were chosen. First it was decided to get analytics from the same kind of template (counties) in different versions of DBpedia, the Portuguese and Japanese. Then, to test the plugins for general datasets, some open COVID-19 datasets were used.

In the first part of this Chapter, a deeper analysis was done in all the selected data. Moreover, a verification of the results was also done, enlightening more information about the findings. Later, a comparison with the selected datasets returned missing properties and different values inside them. Lastly, some additional comments were exposed.

4.1 COUNTIES

Firstly, in this application, the Brazilian counties from the template "Info/Município do Brasil" and the Japanese cities from the template "*日本の市*" were used as examples, leading to a better discovery of strengths and weaknesses from each.

In the beginning, one of the plausible versions for evaluation was DBpedia en, mainly because of its importance to companies as BBC and The New York Times. However, due to the lack of connection between the template and the resources properties, the mentioned version was rejected and replaced by the Japanese version, which seamed complete and had its templates localizable.

Moreover, these were great examples because of the ease in finding the official lists containing the resources they should have. In Brazil this data is available on IBGE[43] whereas the Japanese counties are available in the English version of Wikipedia[44].

4.1.1 Overview

To do a proper comparison between these two versions, some steps were required previously. First, it was essential to understand how the plugins would be connected and which entries they would have to receive. Then, these entries would have to go though some

[43] https://cidades.ibge.gov.br/

[44] https://en.wikipedia.org/wiki/List_of_cities_in_Japan

transformations to adapt to the plugin's inputs, and, finally, the selected plugins would run, returning the expected results.

4.1.2 Templates data

Before the analysis, it is appropriate to discuss the data from the chosen templates.

DBpedia pt has a substantial number of triples, likewise its template "Info/Município do Brasil", with 41 properties and a total of 5562 resources. Meanwhile, the Japanese version is slighter, with 8 properties in the determined template and 790 resources.

These differences will be meaningful later, with the calculated results. Even though the percentage takes into consideration the quantity out of the total, having a smaller amount may imply in a superior data quality and, therefore, a better template completeness.

4.1.3 Getting DBpedia completeness statistics

The first analysis done was to check the completeness of the templates (considering its properties and resources).

Whereas future paths could differ depending on the final destination, the beginning in all cases was the same, get data from DBpedia. Get DBpedia Data, therefore, was the first plugin that had to be called in order to provide the required data for the following steps. Despite working in all situations, it receives different parameters depending on its usage.

4.1.3.1 Get DBpedia Data

The expected inputs to get the completeness of the template resources were the template properties and resources with respective properties. Hence, the plugin should be called twice (figure 28), each call returning one of the desired entries, with the template properties being the first and the resources properties the second.

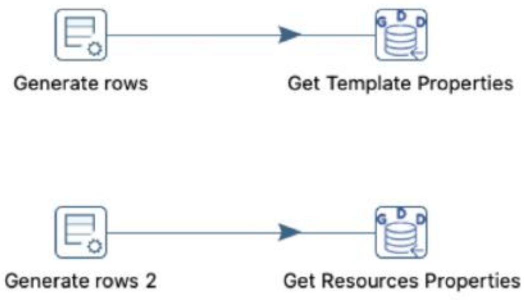


Figure 28 - Get Template Properties in Kettle

The last situation in which it was used was to get a single resource properties, to be used as entry for Resource Properties Analyzer. Therefore, the main distinction this time was that the user had to specify the resource they wanted to evaluate.

4.1.3.2 Primary transformations

Afterwards, the required data had to undergo through some crucial transformations to fit the input standards for the profiling plugins, such as assortment, in case they were not already, and concatenation of fields.

The Get DBpedia Data plugin already sorts the properties before returning them, thereby an assortment was not needed, and this step could be skipped. However, the plugins Property Analyzer and "Resource Property Analyzer" have to receive the template properties as an array of values instead of multiple rows, so a transformation in the returned value was the ensued process.

In order to concatenate a large amount of lines in one the step "Group by" was called (figure 29). It is innate to Kettle and, therefore, no extra installation was required. The plugin receives as input the fields it has to use as keys, the ones to group, and how this grouping should be done.

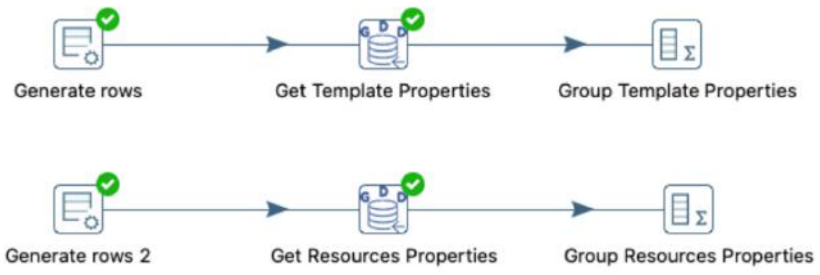


Figure 29 - Group by used to group data previously found

When used with template properties, it groups them by DBpedia version and template name, making it ready to be used as input by other plugins.

On the other hand, when used both with all the resources properties and an individual resource properties, it groups them by resource name, in order to concatenate in one line the properties of each of them.

Later, to merge the two separate fields (template properties and resources properties) into one, the step “Merge join” was called (figure 30), which is also innate to Kettle. It receives the key fields from both tables and the type of join it has to perform, which can either be “Inner join”, “Full outer”, “Left outer” or “Right outer”.

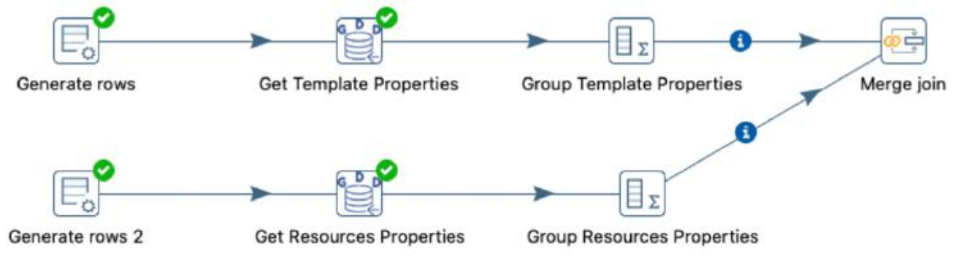


Figure 30 - Merge join used to merge template properties and resources properties

For the first merge join, the chosen type was “Full Outer” and the parameters were the DBpedia version and the template name. The result was a single table with each resource in one line along with their properties and the template properties.

Finally, two profiling plugins were called to get the completeness from both the resources and properties.

4.1.3.3 Template Resource Analyzer

The first statistics generated were from Template Resource Analyzer (figure 31), that gets the resources completeness from a certain template, showing which are most and least complete.

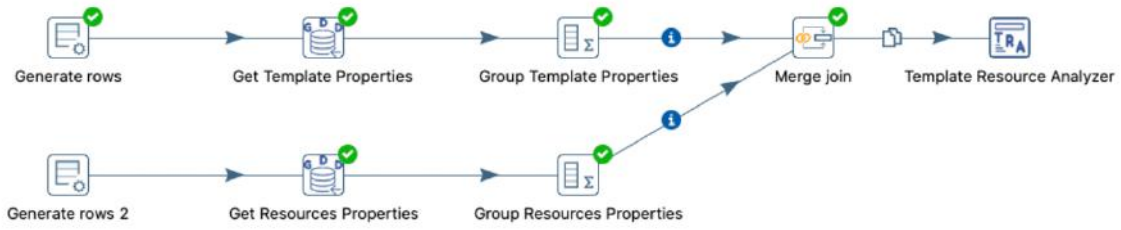
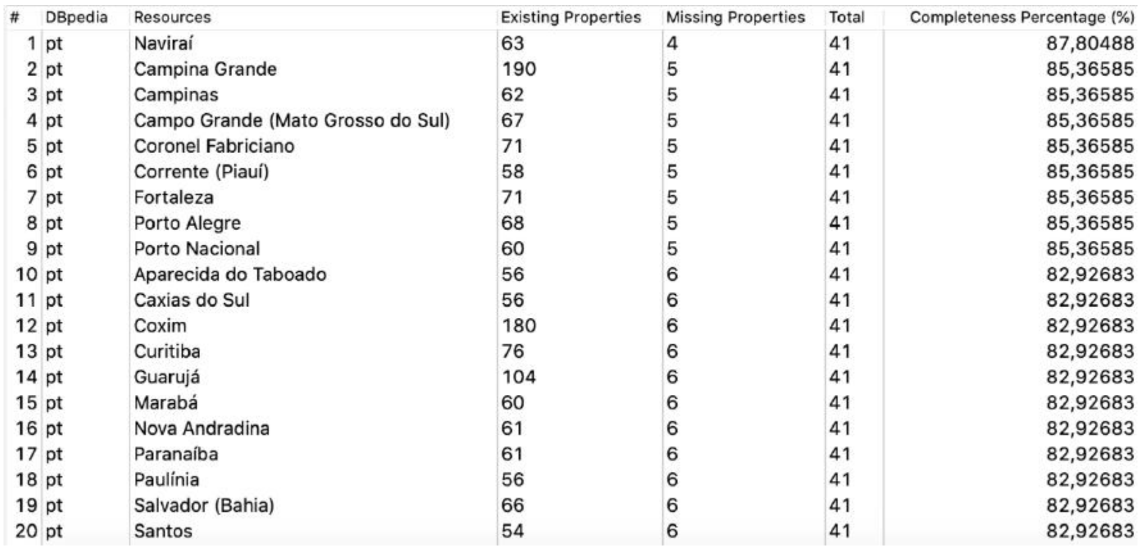


Figure 31 - Template Resource Analyzer used to get resources completeness percentages

As it received as input a table containing the necessary fields, the only need was to relate each of them to the expected entry, for instance, the property list to the input that expects to obtain an array with the template properties, and so on.

When the program ran, it got the resources that belonged to the template “Info/Município do Brasil” with their respective properties and compared them to the template properties, to find the completeness of each and, hence, of the entire template. They were then sorted in descending order (figure 32).



#	DBpedia	Resources	Existing Properties	Missing Properties	Total	Completeness Percentage (%)
1	pt	Naviraí	63	4	41	87,80488
2	pt	Campina Grande	190	5	41	85,36585
3	pt	Campinas	62	5	41	85,36585
4	pt	Campo Grande (Mato Grosso do Sul)	67	5	41	85,36585
5	pt	Coronel Fabriciano	71	5	41	85,36585
6	pt	Corrente (Piauí)	58	5	41	85,36585
7	pt	Fortaleza	71	5	41	85,36585
8	pt	Porto Alegre	68	5	41	85,36585
9	pt	Porto Nacional	60	5	41	85,36585
10	pt	Aparecida do Taboado	56	6	41	82,92683
11	pt	Caxias do Sul	56	6	41	82,92683
12	pt	Coxim	180	6	41	82,92683
13	pt	Curitiba	76	6	41	82,92683
14	pt	Guarujá	104	6	41	82,92683
15	pt	Marabá	60	6	41	82,92683
16	pt	Nova Andradina	61	6	41	82,92683
17	pt	Paranaíba	61	6	41	82,92683
18	pt	Paulínia	56	6	41	82,92683
19	pt	Salvador (Bahia)	66	6	41	82,92683
20	pt	Santos	54	6	41	82,92683

Figure 32 - Resources sorted in descending order by their completeness percentages

Even though some resources had a bigger quantity of properties, it does not mean their completeness should be better, considering they may contain plenty of not mapped properties.

Looking at the numbers above is easy to realize that there is a divergence in the quantity of properties each resource has, going from 88% to 20%.

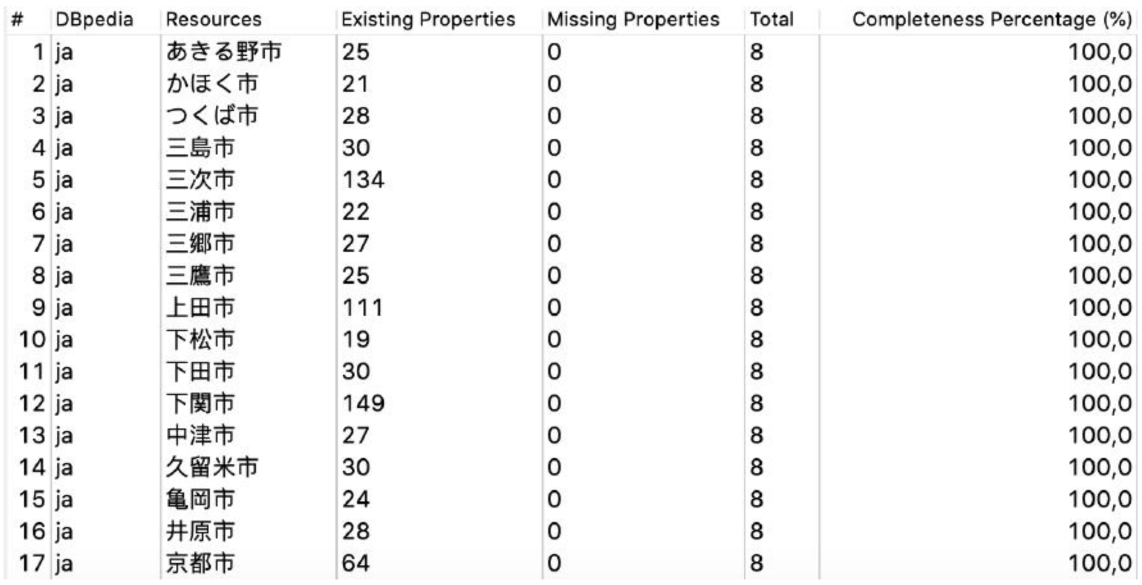
The resource “São Mateus do Sul” had the lowest completeness with almost 20%, and by checking its DBpedia page[45] is visible its incompleteness, with only 10 properties from the template.

The most surprising find was “Naviraí”, a Brazilian county in the state of Mato Grosso do Sul, as being the most complete resource. When looking at its DBpedia page[46] is easy to understand the reason for the highest place. It appears extremely complete, with only 4 template properties missing.

45 http://pt.dbpedia.org/resource/S%C3%A3o_Mateus_do_Sul
46 http://pt.dbpedia.org/resource/Navira%C3%AD

Following, the second analysis was done with the template “日本の市” (Japanese counties), from the Japanese DBpedia.

After finding the resources with their respective percentages, the returned data were sorted (figure 33) and evaluated.



#	DBpedia	Resources	Existing Properties	Missing Properties	Total	Completeness Percentage (%)
1	ja	あきる野市	25	0	8	100,0
2	ja	かほく市	21	0	8	100,0
3	ja	つくば市	28	0	8	100,0
4	ja	三島市	30	0	8	100,0
5	ja	三次市	134	0	8	100,0
6	ja	三浦市	22	0	8	100,0
7	ja	三郷市	27	0	8	100,0
8	ja	三鷹市	25	0	8	100,0
9	ja	上田市	111	0	8	100,0
10	ja	下松市	19	0	8	100,0
11	ja	下田市	30	0	8	100,0
12	ja	下関市	149	0	8	100,0
13	ja	中津市	27	0	8	100,0
14	ja	久留米市	30	0	8	100,0
15	ja	亀岡市	24	0	8	100,0
16	ja	井原市	28	0	8	100,0
17	ja	京都市	64	0	8	100,0

Figure 33 - Japanese resources sorted in descending order

One of the resources that had the highest percentage was “*あきる野市*”[47], Akiruno, a Japanese city. It had all the template properties and, hence, a completeness of 100%, an impressive value. Actually, the result shows that plenty of other resources had the same percentage, indicating how complete the dataset may be.

On the other hand, the resource with the lowest percentage was “*能代市*”[48], Noshiro. It had, however, a completeness of 75%, high above the average.

The report stated that even though some resources did not have all the template properties, they were still very complete, keeping the template percentage high.

While the Japanese version had an abundance of resources with 100% as completeness percentage, in DBpedia pt the highest had almost 88%, a difference of 12% between the highest scores from each version.

The biggest disparity, nonetheless, was with the lowest resources. The worst completeness in the Brazilian version was almost 20%, against 75% in the Japanese. Although there were fewer properties in the Japanese template, it could indicate the attention they give to

[47] http://ja.dbpedia.org/page/%E3%81%82%E3%81%8D%E3%82%8B%E9%87%8E%E5%B8%82
[48] http://ja.dbpedia.org/page/%E8%83%BD%E4%BB%A3%E5%B8%82

their data before including them into the database, leading to a completeness of 97.3% for the Japanese template whereas it is only 58.3% in the Brazilian.

4.1.3.4 Template Property Analyzer

The next statistics were generated by Template Property Analyzer, that gets the properties completeness considering their frequency inside the resources, showing which are most and least complete.

To allow its usage, some previous transformations (figure 34) were necessary, in order to provide the correct input for this plugin. First, the plugin Get DBpedia Data got the resources from the chosen template, followed by "Group by", that concentrated all of them in a single row, aiming their usage as an array in the future. Next, the resources properties delivered by Get DBpedia Data were sorted by the properties names using "Sort rows", and grouped by them, returning for each property a single row with the resources that had them concatenated by a comma.

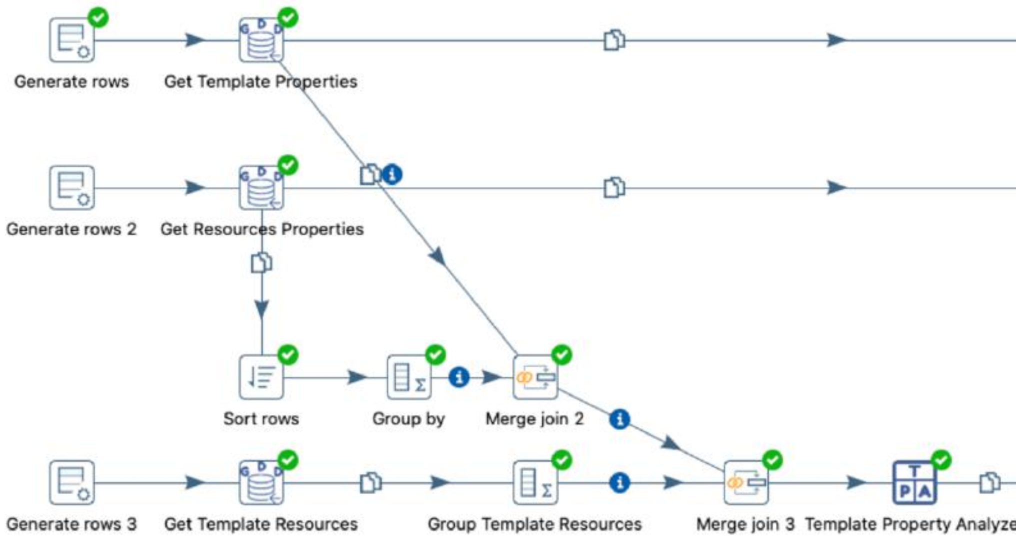
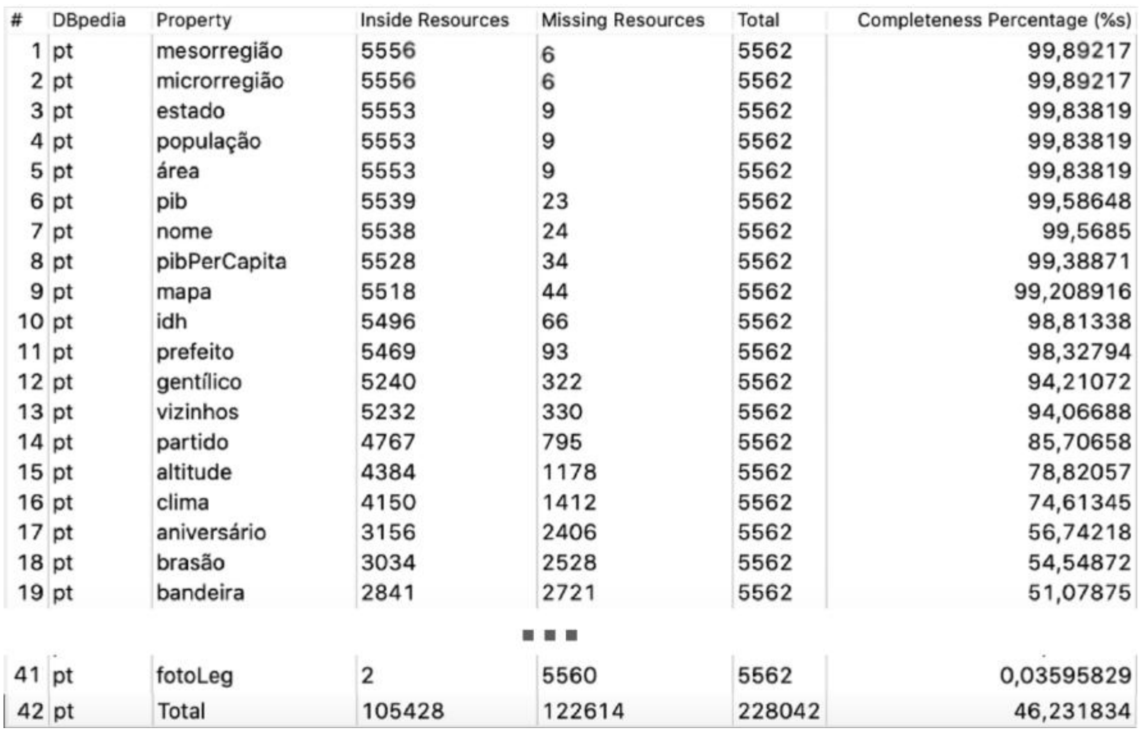


Figure 34 - Necessary transformations to use Template Property Analyzer

Later, the list was merged with the template properties, even the ones that did not belong to any resource, and then merged again with the result of the first "Group by" called. This way, there would be possible to have in the same line the name of the property, the list of resources the property belonged and the complete set of template resources, making the comparison easier.

Basically, to initialize the transformation the user had to select the fields they received as input and place them in their respective places.

The program got the properties from both the resources and the template, to compare them. Later, it returned the properties completeness and sorted them (figure 35).



#	DBpedia	Property	Inside Resources	Missing Resources	Total	Completeness Percentage (%s)
1	pt	mesorregião	5556	6	5562	99,89217
2	pt	microrregião	5556	6	5562	99,89217
3	pt	estado	5553	9	5562	99,83819
4	pt	população	5553	9	5562	99,83819
5	pt	área	5553	9	5562	99,83819
6	pt	pib	5539	23	5562	99,58648
7	pt	nome	5538	24	5562	99,5685
8	pt	pibPerCapita	5528	34	5562	99,38871
9	pt	mapa	5518	44	5562	99,208916
10	pt	idh	5496	66	5562	98,81338
11	pt	prefeito	5469	93	5562	98,32794
12	pt	gentílico	5240	322	5562	94,21072
13	pt	vizinhos	5232	330	5562	94,06688
14	pt	partido	4767	795	5562	85,70658
15	pt	altitude	4384	1178	5562	78,82057
16	pt	clima	4150	1412	5562	74,61345
17	pt	aniversário	3156	2406	5562	56,74218
18	pt	brasão	3034	2528	5562	54,54872
19	pt	bandeira	2841	2721	5562	51,07875
				■ ■ ■		
41	pt	fotoLeg	2	5560	5562	0,03595829
42	pt	Total	105428	122614	228042	46,231834

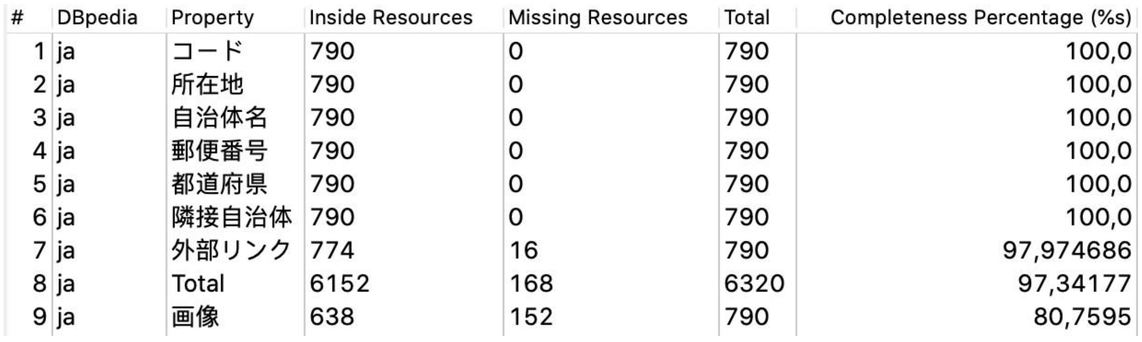
Figure 35 - Properties sorted in descending order by their completeness percentages

The most relevant information it discovered was how the properties were being used in the template, because while some were present in almost all the resources, others were only in 2 out of more than 5000. It gave the template a big disparity in its properties percentages, summing up to a total of 46% as the template completeness.

As shown in figure 85, the best property (mesorregião) was in 5556 out of 5562 resources. It joined the podium with only one more property, microrregião. They were followed by three other properties that did not appear in 9 resources, and lastly, there was the property that only appeared in 2 resources, fotoLeg.

When it comes to the Japanese version, the plugin had to be filled with information taken from the Japanese cities template.

The program did its natural routine, getting the resources and checking the existence of the template properties in each of them. The results were returned and, later, sorted, as shown in figure 36.



#	DBpedia	Property	Inside Resources	Missing Resources	Total	Completeness Percentage (%s)
1	ja	コード	790	0	790	100,0
2	ja	所在地	790	0	790	100,0
3	ja	自治体名	790	0	790	100,0
4	ja	郵便番号	790	0	790	100,0
5	ja	都道府県	790	0	790	100,0
6	ja	隣接自治体	790	0	790	100,0
7	ja	外部リンク	774	16	790	97,974686
8	ja	Total	6152	168	6320	97,34177
9	ja	画像	638	152	790	80,7595

Figure 36 - Sorted properties for Japanese template

The best property was inside 790 out of the 790 resources found on DBpedia, leading to a completeness of 100%. It was the same for 5 other properties. The "worst", however, had a percentage of 80%, adding up to a total completeness of 97% for the template.

The two tests mentioned before had extremely different results when it comes to the highest and lowest completeness, besides the template percentage in general.

While the Japanese template had a perfect percentage in 6 out of 8 properties, representing 75% of the dataset, the Brazilian was missing in 6 resources in its two highest properties. Nevertheless, there were not only 2 properties with a good percentage in DBpedia pt, but, actually, there were 13 properties above 90%, representing 31% of the dataset.

Moreover, the lowest property was also bigger in DBpedia ja, with its 80% against 0.03% in the Portuguese version.

To sum up, even though the quantity of properties in the Japanese version was significantly lower compared to the Portuguese version, the points raised previously led to a better completeness of the Japanese template against the Brazilian.

4.1.3.5 Second transformations

After the first set of analysis was done, it was time to keep looking for more information, using the plugins Property Analyzer and Resource Properties Analyzer. However, to use them properly, some other transformations were necessary.

Firstly, it was decided that these plugins would be called with the highest values found on the previous steps. In addition, intending to get the best property/resource, the plugin "Sample rows" was used, getting only the first line of the streams.

Then, two separate ways were created (figure 37):

1. In the Property Analyzer branch, the value returned by the previous step (property with highest completeness) was merged with the resources properties (delivered by the step “Group by”).
2. In the Resource Properties Analyzer branch, the process was heavier. The past result (resource with highest percentage) was first merged with empty generated rows (to increase the number of available rows) and then it was used to get the resource properties (delivered by Get DBpedia Data). Following, these properties were grouped by the resource name and merged with the template properties, resulting in a row with the resource name, its properties and the template properties.

Finally, the plugins Resource Properties Analyzer and Property Analyzer had the chance to run with their respective entries.

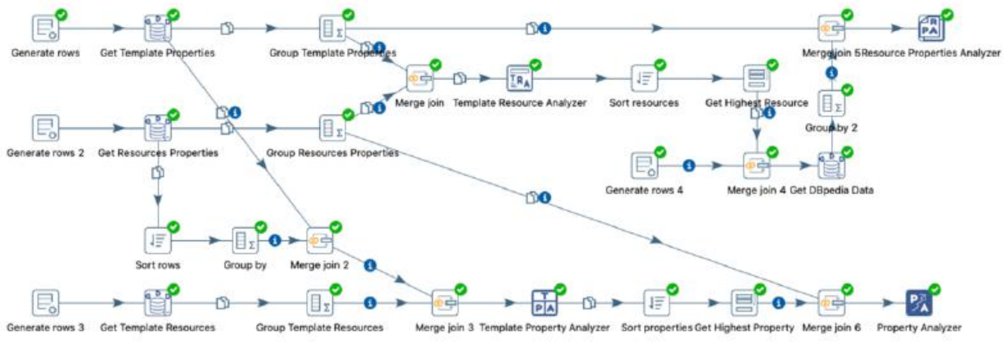


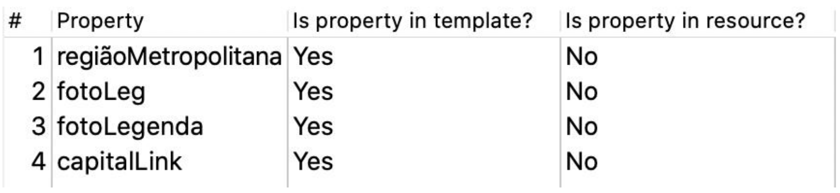
Figure 37 - Kettle entire flow diagram

4.1.3.6 Resource Properties Analyzer

The Resource Properties Analyzer was called with the best resource. This plugin is valuable because it can provide the necessary information to potentially make a resource complete.

For the Brazilian template, the county evaluated was “Naviraí”, the most complete resource.

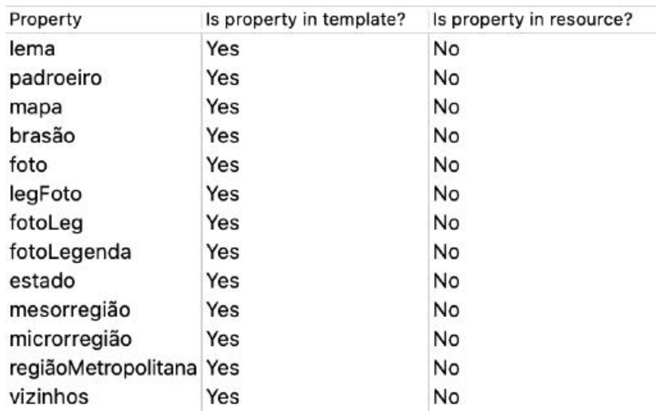
When the program ran, it got the resource properties and compared them to the template properties, making it possible to know which of them were only on the template (figure 38).



#	Property	Is property in template?	Is property in resource?
1	regiãoMetropolitana	Yes	No
2	fotoLeg	Yes	No
3	fotoLegenda	Yes	No
4	capitalLink	Yes	No

Figure 38 - Resource Properties Analyzer result

On top of that, a good contrast to illustrate the huge difference between a complete and an almost empty resource would be to show the results of the same plugin for the county "São Mateus do Sul" (figure 39).



Property	Is property in template?	Is property in resource?
lema	Yes	No
padroeiro	Yes	No
mapa	Yes	No
brasão	Yes	No
foto	Yes	No
legFoto	Yes	No
fotoLeg	Yes	No
fotoLegenda	Yes	No
estado	Yes	No
mesorregião	Yes	No
microrregião	Yes	No
regiãoMetropolitana	Yes	No
vizinhos	Yes	No

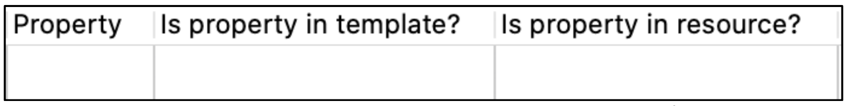
Figure 39 - Resource Properties Analyzer's spreadsheet with São Mateus do Sul' missing properties

The county used in the first example only had 4 missing properties, while the second had over 30. For the sake of researches, the first example would allow a better insight and would help scientists to find better relations navigating through linked data.

Therefore, an impressive use of this plugin would be to improve the resources found, as now their missing properties are visible, making them as complete as possible.

Later, for the Japanese cities, the tested resource was Akiruno, once again, the most complete.

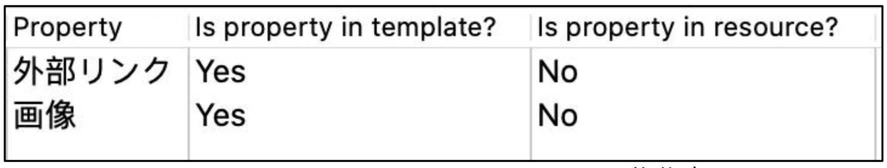
After the entries were inputted, the program did its traditional processing, calculating the percentage based on the comparison between the properties from the resource and the template. The results, later, proved how complete the resource was, with no template property having been left out, hence, a completeness of 100% (figure 40).



Property	Is property in template?	Is property in resource?

Figure 40 - Resource Properties Analyzer's spreadsheet with 武雄市' missing properties

Furthermore, the worst resource of this template, Noshiro, had only 2 missing properties (figure 41), leading to a percentage of 75%.



Property	Is property in template?	Is property in resource?
外部リンク	Yes	No
画像	Yes	No

Figure 41 - Resource Properties Analyzer's spreadsheet with 能代市' missing properties

With these results, it was clear to notice that the difference between the quantity of missing properties from the most complete to the most incomplete was only 2, putting the Japanese DBpedia in a high standard of completeness level.

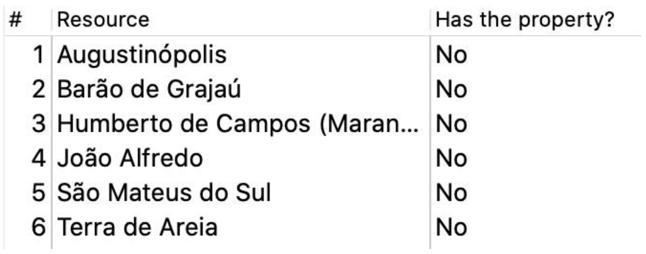
Finally, although the quantity of missing properties of DBpedia ja was fewer, the template itself was too, as described in the section 4.1.2 and, therefore, it was easier for a resource to be fuller.

4.1.3.7 Property Analyzer

The Property Analyzer plugin has the intention of giving the information of which template resources (do not) have the chosen property. This is interesting when used along with Template Properties Analyzer because it gives the user the chance of, after perceiving the best properties, analyze which resources do not have them, aiming to fix it in the future. For instance, if there is a property not used by 2 resources it is easier to know them and fix more immediately.

For this test the property "mesorregião" was used, as it is one of the best properties and would be great to find the resources that do not have it.

As the program started to run, it got all the template resources with their properties and checked if the chosen was among them, returning the ones on which it was missing (figure 42).



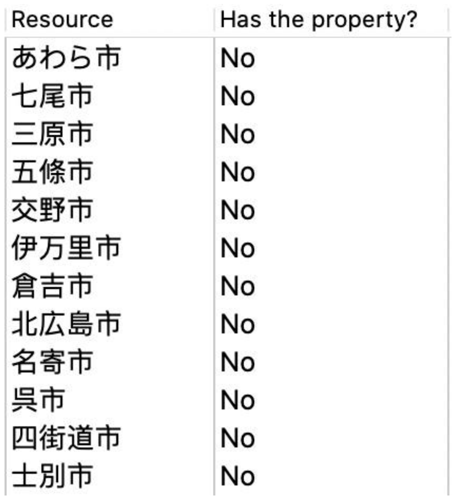
#	Resource	Has the property?
1	Augustinópolis	No
2	Barão de Grajaú	No
3	Humberto de Campos (Maran...	No
4	João Alfredo	No
5	São Mateus do Sul	No
6	Terra de Areia	No

Figure 42 - Property Analyzer's spreadsheet with resources that do not have "mesorregião"

The result matched the information found in Template Property Analyzer, returning the same number in both of them. Furthermore, it is unquestionable that the property had an excellent completeness, considering it was only missing in 6 out of more than 5000 resources.

Similarly, another relevant discovery was the property "コード" (area code), the best in this template.

The program ran the same way as explained before, returning the resources that did not have the wanted property. Moreover, as it was expected, the property had a completeness percentage of 100%, which means that all the resources had it and, therefore, none of them were outputted in the result. However, when it comes to the worst property, *画像* (foaf:img), the result was different (figure 43).



Resource	Has the property?
あわら市	No
七尾市	No
三原市	No
五條市	No
交野市	No
伊万里市	No
倉吉市	No
北広島市	No
名寄市	No
呉市	No
四街道市	No
士別市	No

Figure 43 - Property Analyzer's spreadsheet with resources where "画像" is not inside

Although the property was in an abundance of resources, it was also missing in more than 100, leading to the worst collocation among the template properties.

As seen before, the Japanese template had less properties, but better percentages in each of them. On the other hand, the Brazilian version had plenty of properties, but could not keep them fully complete.

When comparing them side by side, while the worst property of the Portuguese template was only in less than 1% of the resources, the Japanese was in more than 80% of the dataset, corroborating to the points mentioned before.

4.1.4 Checking the template resources

Another analysis done using DBpedia countries was to find the template completeness, by checking the resources it should have and has.

To provide this statistic the plugin Template Resource Input Analyzer was used along with Get DBpedia Data (figure 44).

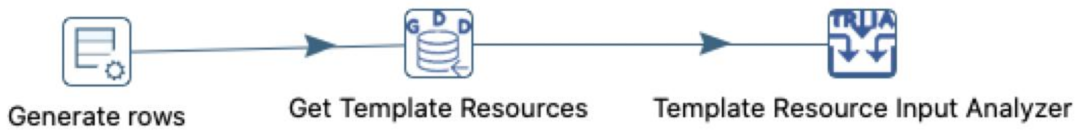


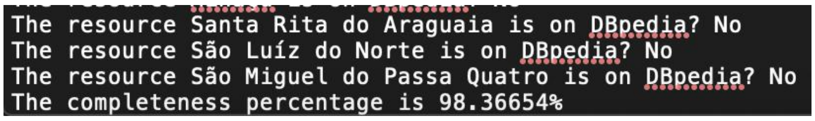
Figure 44 - Template Resource Input Analyzer in Kettle workflow

To begin with, the Get DBpedia Data plugin was called aiming to get the template resources (counties from Brazil and Japan). With the expected return, it was then ensued by the plugin Template Resource Input Analyzer, that got the missing resources compared to the lists provided by the governments.

4.1.4.1 Template Resource Input Analyzer

Initially, to start the analysis with the Brazilian counties, it was needed to get the official data from the IBGE website[49], since the intention was to find the most accurate information of which resources were missing.

Next, it compared them to the ones from the template, returning which were missing on DBpedia (figure 45). Moreover, as usual, the plugin returned the results in a CSV and a report. In this case, opposite to the other examples, the report will be shown, considering it shows not only the resources that are missing but also the template completeness.



```
The resource Santa Rita do Araguaia is on DBpedia? No
The resource São Luíz do Norte is on DBpedia? No
The resource São Miguel do Passa Quatro is on DBpedia? No
The completeness percentage is 98.36654%
```

Figure 45 - Template Resource Input Analyzer's report file with Brazilian counties completeness percentage

However, some issues appeared while the program was running, which will be explained shortly.

Although there were 90 resources missing, it only represents a low number considering there are more than 5000 counties in Brazil. Besides, there was a huge amount of resources that appeared in the list as missing but were in DBpedia.

When analyzing the list of resources that theoretically were not in DBpedia and typing their names in the domain[50], it was possible to realize that plenty of them were not missing indeed.

There were several reasons for this error:

1. The resources names were misspelled. In this kind of error, the resource had a different spelling in DBpedia and in the official CSV, for instance, Açu. In the IBGE list it was written "Açu", while in DBpedia it was "Assu". This problem did not occur only with this word, but, as a matter of fact, it happened a lot in the dataset. It had some variations

[49] https://cidades.ibge.gov.br/
[50] http://pt.dbpedia.org/resource/

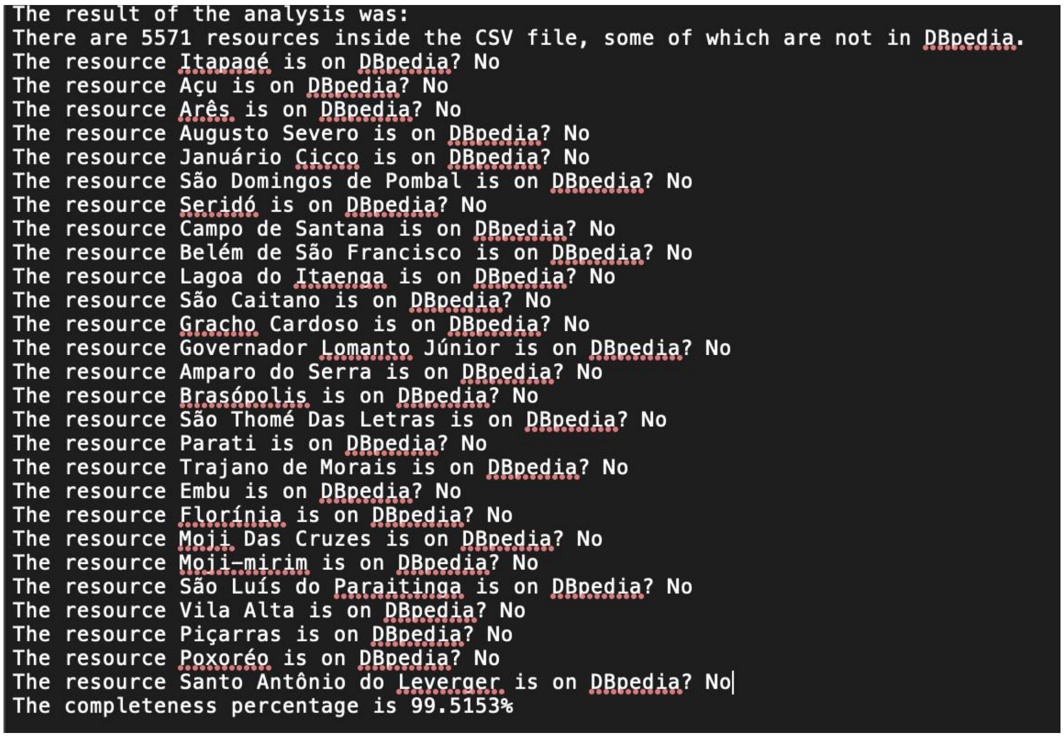
too, because sometimes, although the name was written in the same way, it had a grammatical accent in one of them that was not in the other, or even a punctuation mark.

2. Resources that were not mapped in the template but exists in the dataset. There were resources that existed in DBpedia but were not mapped in the template, making it difficult to find them in the repository.

   To solve this issue, a possible alternative would be to fetch the name in a query. However, it would be complicated to check if the resource was referring to the one in the CSV or another with a similar name.

To try to soften these problems, especially because some resources in DBpedia have the state along with their names, for example, "Coimbra (Minas Gerais)", the software does a validation and removes everything that comes after the parenthesis, besides putting everything in lower case, removing spaces, accents and punctuation that could complicate the comparison, making checking the names as truthful as possible.

Later, a new test was done considering all those grammatical corrections and also looking for resources that belonged to the template but were not mapped there.



```
The result of the analysis was:
There are 5571 resources inside the CSV file, some of which are not in DBpedia.
The resource Itapagé is on DBpedia? No
The resource Açu is on DBpedia? No
The resource Arês is on DBpedia? No
The resource Augusto Severo is on DBpedia? No
The resource Januário Cicco is on DBpedia? No
The resource São Domingos de Pombal is on DBpedia? No
The resource Seridó is on DBpedia? No
The resource Campo de Santana is on DBpedia? No
The resource Belém de São Francisco is on DBpedia? No
The resource Lagoa do Itaenga is on DBpedia? No
The resource São Caitano is on DBpedia? No
The resource Gracho Cardoso is on DBpedia? No
The resource Governador Lomanto Júnior is on DBpedia? No
The resource Amparo do Serra is on DBpedia? No
The resource Brasópolis is on DBpedia? No
The resource São Thomé Das Letras is on DBpedia? No
The resource Parati is on DBpedia? No
The resource Trajano de Morais is on DBpedia? No
The resource Embu is on DBpedia? No
The resource Florínia is on DBpedia? No
The resource Moji Das Cruzes is on DBpedia? No
The resource Moji-mirim is on DBpedia? No
The resource São Luís do Paraitinga is on DBpedia? No
The resource Vila Alta is on DBpedia? No
The resource Piçarras is on DBpedia? No
The resource Poxoréo is on DBpedia? No
The resource Santo Antônio do Leverger is on DBpedia? No
The completeness percentage is 99.5153%
```

Figure 46 - Template Resource Input Analyzer's report file with not mapped Brazilian counties that are not in DBpedia

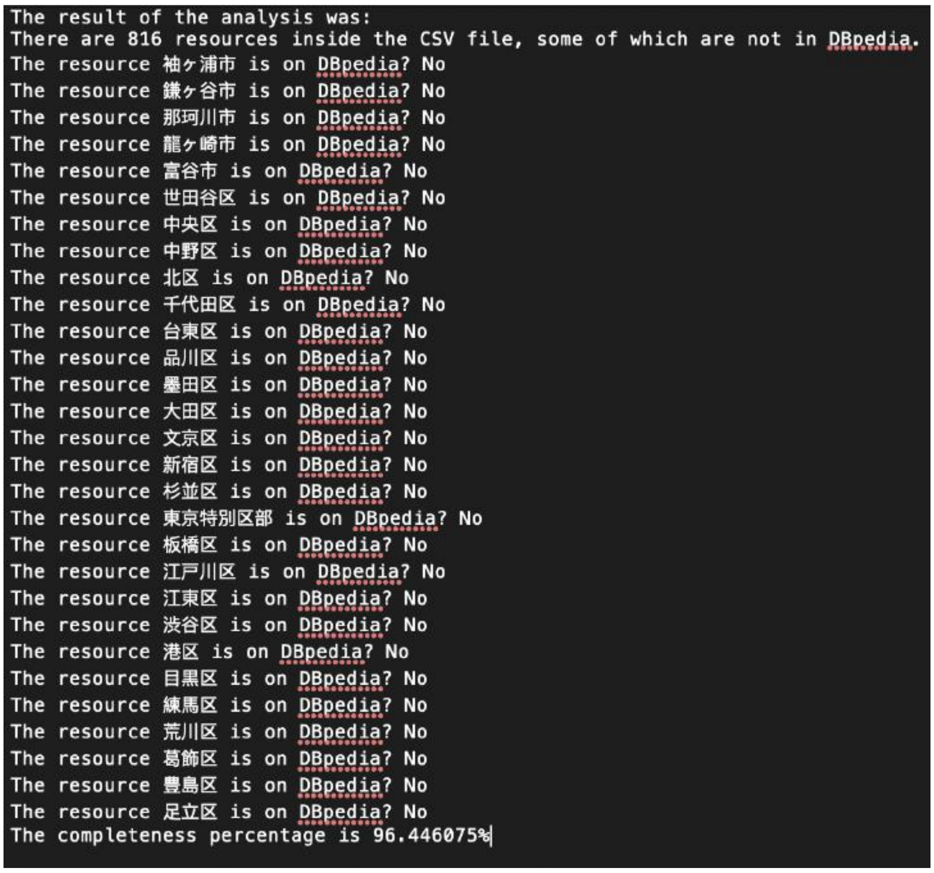
With the results (figure 46) it was possible to realize that the changes mentioned helped to identify more resources, with now only 27 resources missing out of more than 5000. Nevertheless, there were some resources that still couldn't be found, mostly because of wrong letters, or because neither they were mapped in the template nor they were in its list.

For the Japanese comparison, the list of cities from Japan was taken from the English Wikipedia[51]. They were cautiously copied into a CSV file and used by the plugin as input.

It is also important to mention that an official data was searched meticulously, but neither they were in a good format to use nor they were all in the same page. Then, it was concluded that Wikipedia's data was the most complete and of easy access, hence, becoming the main source of comparison for this test.

The program got all the Japanese cities mapped in the template and compare to the ones extracted from Wikipedia, to find the template completeness.

The data went through the same process as mentioned before (removing parenthesis and putting everything in lower case), and the plugin finally returned the missing resources (figure 47).



```
The result of the analysis was:
There are 816 resources inside the CSV file, some of which are not in DBpedia.
The resource 袖ヶ浦市 is on DBpedia? No
The resource 鎌ヶ谷市 is on DBpedia? No
The resource 那珂川市 is on DBpedia? No
The resource 龍ヶ崎市 is on DBpedia? No
The resource 富谷市 is on DBpedia? No
The resource 世田谷区 is on DBpedia? No
The resource 中央区 is on DBpedia? No
The resource 中野区 is on DBpedia? No
The resource 北区 is on DBpedia? No
The resource 千代田区 is on DBpedia? No
The resource 台東区 is on DBpedia? No
The resource 品川区 is on DBpedia? No
The resource 墨田区 is on DBpedia? No
The resource 大田区 is on DBpedia? No
The resource 文京区 is on DBpedia? No
The resource 新宿区 is on DBpedia? No
The resource 杉並区 is on DBpedia? No
The resource 東京特別区部 is on DBpedia? No
The resource 板橋区 is on DBpedia? No
The resource 江戸川区 is on DBpedia? No
The resource 江東区 is on DBpedia? No
The resource 渋谷区 is on DBpedia? No
The resource 港区 is on DBpedia? No
The resource 目黒区 is on DBpedia? No
The resource 練馬区 is on DBpedia? No
The resource 荒川区 is on DBpedia? No
The resource 葛飾区 is on DBpedia? No
The resource 豊島区 is on DBpedia? No
The resource 足立区 is on DBpedia? No
The completeness percentage is 96.446075%
```

Figure 47 - Template Resource Input Analyzer's report file with Japanese cities that are not in DBpedia and completeness percentage

[51] https://en.wikipedia.org/wiki/List_of_cities_in_Japan

There were 29 resources missing in DBpedia from a total of 816, in which plenty of them were actually there but were not mapped in the template. This problem could also be solved by trying to fetch the resources directly, but, once again, it would not ensure they were the real data the user was looking for.

Although in the last results the Japanese version was having a better ranking than the Brazilian, DBpedia pt took an advantage this time, raising its completeness to 98% against 96% of DBpedia ja.

However, it is also a possibility that both of them have all the resources in their datasets, but either they were not mapped in the template (which can be considered a mapping problem) or were misspelled.

4.1.4.2 Resource Input Analyzer

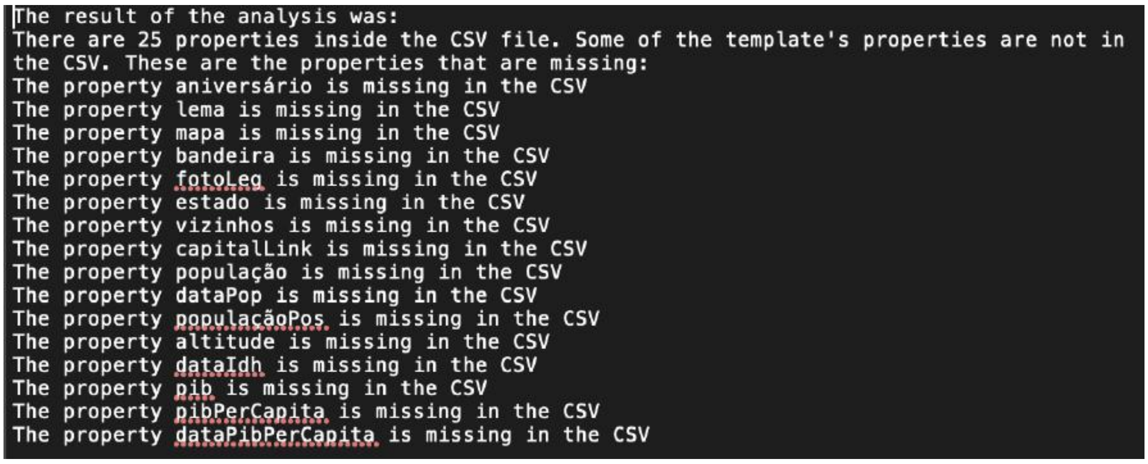
As an extra step, Resource Input Analyzer comes with the intention of verifying missing properties in a CSV input in comparison to a template in DBpedia. As it was mentioned in chapter 3, it may be simple but at the same time really helpful as it was created aiming to assist in the insertion of fuller resources into the repository. Therefore, it analyzes all the properties inputted and returns the template properties that are missing.

Although this plugin has 3 outputs, the one chosen to represent these executions was the report, as it shows the quantity of properties inside the file, besides which are missing.

To test this plugin a mock file[52] was created with some of the template properties. Moreover, the properties were discovered by the Web scraper and used as input.

The program compared the properties in the CSV file with the ones found inside the template and returned which were missing (figure 48). This way, the user could input the properties from a resource they are creating and check if they are complete.

[52] https://github.com/ingridpacheco/ETL4Profiling/blob/master/usage/COUNTIES/Brazilian_counties_properties.xlsx

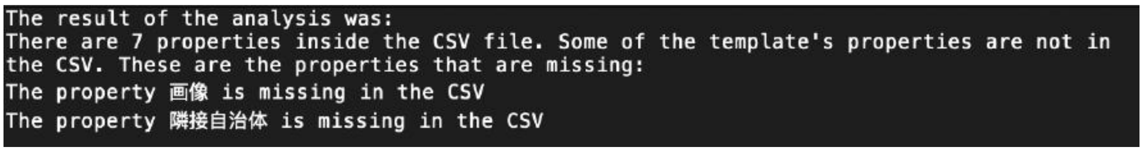


```
The result of the analysis was:
There are 25 properties inside the CSV file. Some of the template's properties are not in
the CSV. These are the properties that are missing:
The property aniversário is missing in the CSV
The property lema is missing in the CSV
The property mapa is missing in the CSV
The property bandeira is missing in the CSV
The property fotoLeg is missing in the CSV
The property estado is missing in the CSV
The property vizinhos is missing in the CSV
The property capitalLink is missing in the CSV
The property população is missing in the CSV
The property dataPop is missing in the CSV
The property populaçãoPos is missing in the CSV
The property altitude is missing in the CSV
The property dataIdh is missing in the CSV
The property pib is missing in the CSV
The property pibPerCapita is missing in the CSV
The property dataPibPerCapita is missing in the CSV
```

Figure 48 - Resource Input Analyzer’s report file with Brazilian counties properties that are missing in the input

To test the Japanese cities another mock file[53] was created, but this time using some properties from the respective template. Two of them were missing, the ones with lowest and highest completeness.

When the program ran (figure 49), it fetched the template properties and compared them to the ones from the CSV file. It, then, alerted which properties were missing, as expected.



```
The result of the analysis was:
There are 7 properties inside the CSV file. Some of the template's properties are not in
the CSV. These are the properties that are missing:
The property 画像 is missing in the CSV
The property 隣接自治体 is missing in the CSV
```

Figure 49 - Resource Input Analyzer’s report file with Japanese cities’ properties that are missing in the input

In this plugin there is not much to compare, because it only returns the missing properties as it was meant to. Moreover, it was designed to be used in the creation of a new resource, instead of the assessment of old ones.

4.1.5 Completeness between resources

Lastly, another analysis done was a general check of DBpedia resources considering the completeness of its properties. While the Template Resource Analyzer evaluates resources comparing them to the template properties, this analysis was focused on the fuller context of resource properties. Each resource has an impressive amount of properties, which can be either inside the template or only in their scope. This test checked the resource completeness considering all the properties that exist in the dataset, even only inside others.

[53] https://github.com/ingridpacheco/ETL4Profiling/blob/master/usage/COUNTIES/Japanese_cities_properties.xlsx

Even though some properties did not belong to the template, they could increase the data quality and, therefore, could be considered important for every resource.

This analysis happened in two steps. First, the DBpedia data was fetched and triplified using DBpedia Triplification. After, the plugin Inner Profiling was called (figure 50), to evaluate the missing properties for each resource.

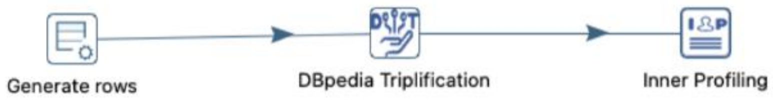
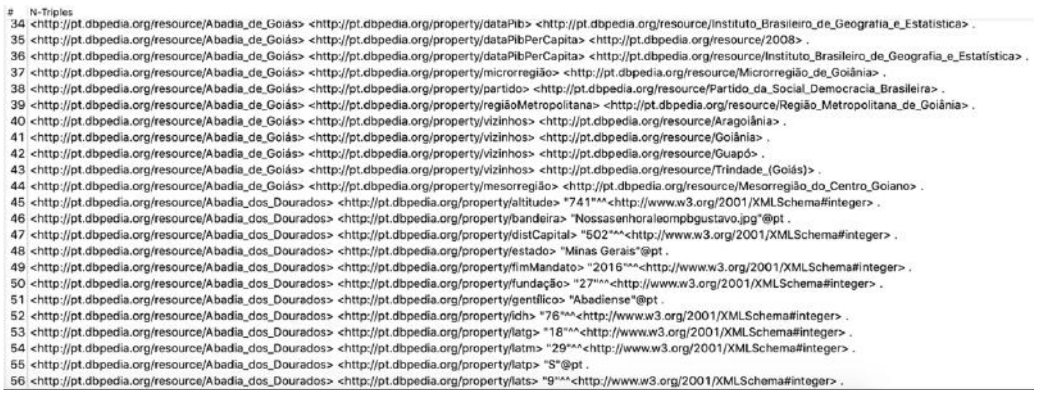


Figure 50 - Flow to get completeness between resources

4.1.5.1 DBpedia Triplification

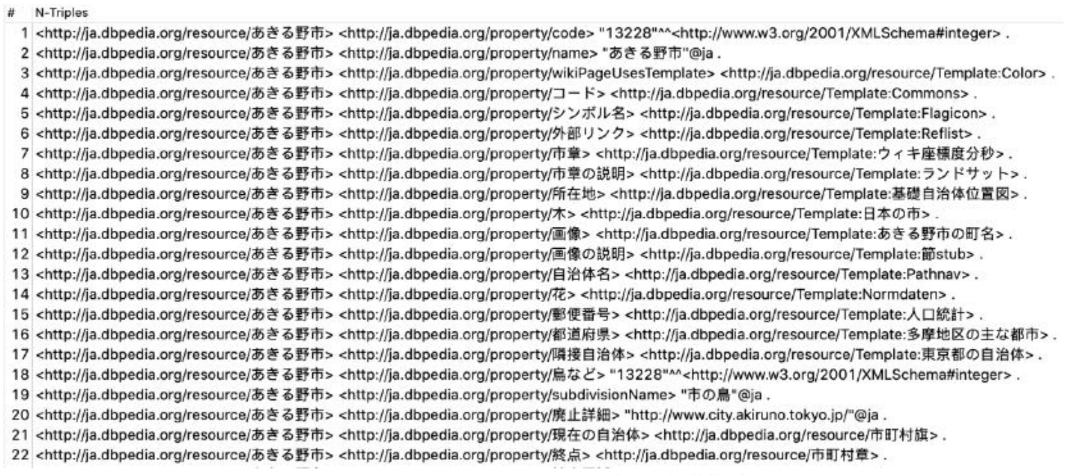
The DBpedia Triplification was done due to the need of using DBpedia data in future comparisons with other datasets or even for an inner profiling. It fetches the template chosen and triplifies the information found, with subject (resource), predicate (properties) and value, following the rules presented in chapter 3.

As another comparison between counties was intended, it was set with both the Brazilian (figure 51) and Japanese (figure 52) templates.



```
N-Triples
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/dataPib> <http://pt.dbpedia.org/resource/Instituto_Brasileiro_de_Geografia_e_Estatistica> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/dataPibPerCapita> <http://pt.dbpedia.org/resource/2008> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/dataPibPerCapita> <http://pt.dbpedia.org/resource/Instituto_Brasileiro_de_Geografia_e_Estatística> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/microrregião> <http://pt.dbpedia.org/resource/Microrregião_de_Goiânia> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/partido> <http://pt.dbpedia.org/resource/Partido_da_Social_Democracia_Brasileira> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/regiãoMetropolitana> <http://pt.dbpedia.org/resource/Região_Metropolitana_de_Goiânia> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/vizinhos> <http://pt.dbpedia.org/resource/Aragoiânia> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/vizinhos> <http://pt.dbpedia.org/resource/Goiânia> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/vizinhos> <http://pt.dbpedia.org/resource/Guapó> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/vizinhos> <http://pt.dbpedia.org/resource/Trindade_(Goiás)> .
<http://pt.dbpedia.org/resource/Abadia_de_Goiás> <http://pt.dbpedia.org/property/mesorregião> <http://pt.dbpedia.org/resource/Mesorregião_do_Centro_Goiano> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/altitude> "741"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/bandeira> "Nossasenhoraleompbgustavo.jpg"@pt .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/distCapital> "502"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/estado> "Minas Gerais"@pt .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/fimMandato> "2016"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/fundação> "27"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/gentílico> "Abadiense"@pt .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/idh> "76"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/latg> "18"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/latm> "29"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/latp> "S"@pt .
<http://pt.dbpedia.org/resource/Abadia_dos_Dourados> <http://pt.dbpedia.org/property/lats> "9"^^<http://www.w3.org/2001/XMLSchema#integer> .
```

Figure 51 - Brazilian counties triplified



```
#  N-Triples
 1 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/code> "13228"^^<http://www.w3.org/2001/XMLSchema#integer> .
 2 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/name> "あきる野市"@ja .
 3 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/wikiPageUsesTemplate> <http://ja.dbpedia.org/resource/Template:Color> .
 4 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/コード> <http://ja.dbpedia.org/resource/Template:Commons> .
 5 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/シンボル名> <http://ja.dbpedia.org/resource/Template:Flagicon> .
 6 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/外部リンク> <http://ja.dbpedia.org/resource/Template:Reflist> .
 7 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/市章> <http://ja.dbpedia.org/resource/Template:ウィキ座標度分秒> .
 8 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/市章の説明> <http://ja.dbpedia.org/resource/Template:ランドサット> .
 9 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/所在地> <http://ja.dbpedia.org/resource/Template:基礎自治体位置図> .
10 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/木> <http://ja.dbpedia.org/resource/Template:日本の市> .
11 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/画像> <http://ja.dbpedia.org/resource/Template:あきる野市の町名> .
12 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/画像の説明> <http://ja.dbpedia.org/resource/Template:節stub> .
13 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/自治体名> <http://ja.dbpedia.org/resource/Template:Pathnav> .
14 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/花> <http://ja.dbpedia.org/resource/Template:Normdaten> .
15 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/郵便番号> <http://ja.dbpedia.org/resource/Template:人口統計> .
16 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/都道府県> <http://ja.dbpedia.org/resource/Template:多摩地区の主な都市> .
17 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/隣接自治体> <http://ja.dbpedia.org/resource/Template:東京都の自治体> .
18 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/鳥など> "13228"^^<http://www.w3.org/2001/XMLSchema#integer> .
19 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/subdivisionName> "市の鳥"@ja .
20 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/廃止詳細> "http://www.city.akiruno.tokyo.jp/"@ja .
21 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/現在の自治体> <http://ja.dbpedia.org/resource/市町村旗> .
22 <http://ja.dbpedia.org/resource/あきる野市> <http://ja.dbpedia.org/property/終点> <http://ja.dbpedia.org/resource/市町村章> .
```

Figure 52 - Japanese counties triplified

Firstly, the plugin got all the template resources by running a query. Later, it searched, through a Web scrapping, its properties/values and did peculiar alterations, returning the correct triplification.

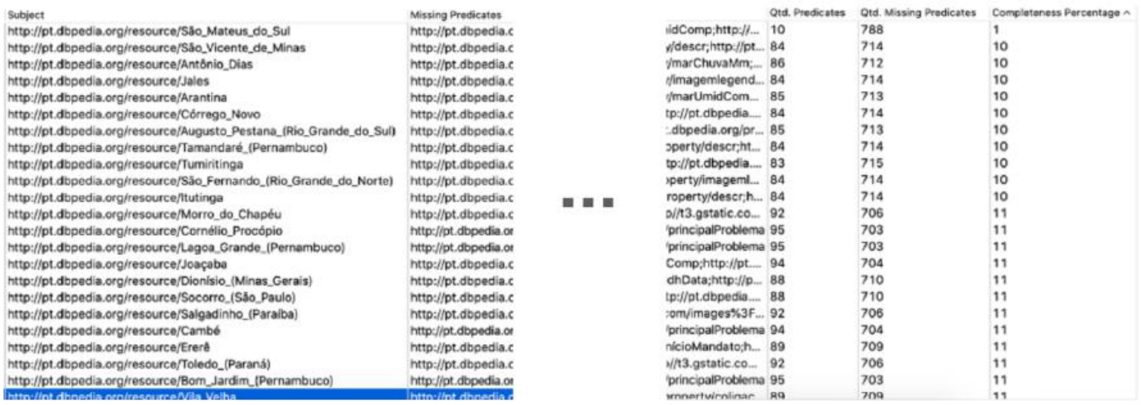
Lastly, after all the processes were done, it provided the triples in both a CSV file and Pentaho's fields, allowing it to be used as input in future plugins. When it comes to Pentaho's spreadsheet, it returned the resources with their respective properties, values and types in a single field named "N-Triples", which was later used by Inner Profiling.

4.1.5.2 Inner Profiling

The Inner Profiling plugin makes comparisons inside any dataset provided by the user. It has the objective of knowing which subjects are more complete than others, and which predicates are missing in each of them.

Using the triplified data from the last plugin a comparison was done indicating which counties had bigger/lower percentages.

It returned the results (figure 53) containing the missing predicates for all the subjects and their respective completeness, calculated based on the total number of predicates found in the dataset.



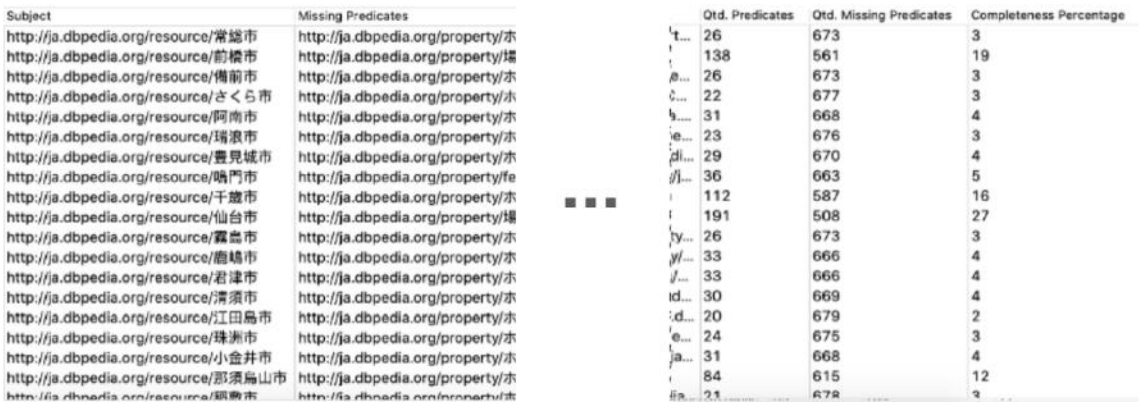
Subject	Missing Predicates		Qtd. Predicates	Qtd. Missing Predicates	Completeness Percentage ^
http://pt.dbpedia.org/resource/São_Mateus_do_Sul	http://pt.dbpedia.c	idComp;http://...	10	788	1
http://pt.dbpedia.org/resource/São_Vicente_de_Minas	http://pt.dbpedia.c	y/descr;http://pt...	84	714	10
http://pt.dbpedia.org/resource/Antônio_Dias	http://pt.dbpedia.c	/marChuvaMm;...	86	712	10
http://pt.dbpedia.org/resource/Jales	http://pt.dbpedia.c	/imagemlegend...	84	714	10
http://pt.dbpedia.org/resource/Arantina	http://pt.dbpedia.c	/marUmidCom...	85	713	10
http://pt.dbpedia.org/resource/Córrego_Novo	http://pt.dbpedia.c	tp://pt.dbpedia....	84	714	10
http://pt.dbpedia.org/resource/Augusto_Pestana_(Rio_Grande_do_Sul)	http://pt.dbpedia.c	.dbpedia.org/pr...	85	713	10
http://pt.dbpedia.org/resource/Tamandaré_(Pernambuco)	http://pt.dbpedia.c	operty/descr;ht...	84	714	10
http://pt.dbpedia.org/resource/Tumiritinga	http://pt.dbpedia.c	tp://pt.dbpedia....	83	715	10
http://pt.dbpedia.org/resource/São_Fernando_(Rio_Grande_do_Norte)	http://pt.dbpedia.c	operty/imageml...	84	714	10
http://pt.dbpedia.org/resource/Itutinga	http://pt.dbpedia.c	roperty/descr;h...	84	714	10
http://pt.dbpedia.org/resource/Morro_do_Chapéu	http://pt.dbpedia.c	o//t3.gstatic.co...	92	706	11
http://pt.dbpedia.org/resource/Cornélio_Procópio	http://pt.dbpedia.or	'principalProblema	95	703	11
http://pt.dbpedia.org/resource/Lagoa_Grande_(Pernambuco)	http://pt.dbpedia.or	'principalProblema	95	703	11
http://pt.dbpedia.org/resource/Joaçaba	http://pt.dbpedia.c	Comp;http://pt....	94	704	11
http://pt.dbpedia.org/resource/Dionísio_(Minas_Gerais)	http://pt.dbpedia.c	dhData;http://p...	88	710	11
http://pt.dbpedia.org/resource/Socorro_(São_Paulo)	http://pt.dbpedia.c	tp://pt.dbpedia....	88	710	11
http://pt.dbpedia.org/resource/Salgadinho_(Paraíba)	http://pt.dbpedia.c	om/images%3F...	92	706	11
http://pt.dbpedia.org/resource/Cambé	http://pt.dbpedia.or	'principalProblema	94	704	11
http://pt.dbpedia.org/resource/Ererê	http://pt.dbpedia.c	nícioMandato;h...	89	709	11
http://pt.dbpedia.org/resource/Toledo_(Paraná)	http://pt.dbpedia.c	//t3.gstatic.co...	92	706	11
http://pt.dbpedia.org/resource/Bom_Jardim_(Pernambuco)	http://pt.dbpedia.or	'principalProblema	95	703	11
http://pt.dbpedia.org/resource/Vila_Velha	http://pt.dbpedia.c	roperty/coligac	89	709	11

Figure 53 - Inner Profiling showing missing predicates for Brazilian template

With the output it was noticeable that the completeness considering all possible properties was very low. This happened because the resources had plenty of properties that were not mapped in the template, as seen previously in some plugins.

The highest percentage was 23% in 3 resources (Dourados, Campina Grande and Juiz de Fora), against 1% as the lowest in 1 resource, São Mateus do Sul.

Following, to calculate the Japanese counties completeness, the resources from the template were fetched, triplified and used as input for this plugin (figure 54).



Subject	Missing Predicates		Qtd. Predicates	Qtd. Missing Predicates	Completeness Percentage
http://ja.dbpedia.org/resource/常総市	http://ja.dbpedia.org/property/ホ	't...	26	673	3
http://ja.dbpedia.org/resource/前橋市	http://ja.dbpedia.org/property/場		138	561	19
http://ja.dbpedia.org/resource/備前市	http://ja.dbpedia.org/property/ホ	e...	26	673	3
http://ja.dbpedia.org/resource/さくら市	http://ja.dbpedia.org/property/ホ	c...	22	677	3
http://ja.dbpedia.org/resource/阿南市	http://ja.dbpedia.org/property/ホ	b....	31	668	4
http://ja.dbpedia.org/resource/瑞浪市	http://ja.dbpedia.org/property/ホ	ie...	23	676	3
http://ja.dbpedia.org/resource/豊見城市	http://ja.dbpedia.org/property/ホ	di...	29	670	4
http://ja.dbpedia.org/resource/鳴門市	http://ja.dbpedia.org/property/fe	//j...	36	663	5
http://ja.dbpedia.org/resource/千歳市	http://ja.dbpedia.org/property/ホ		112	587	16
http://ja.dbpedia.org/resource/仙台市	http://ja.dbpedia.org/property/場		191	508	27
http://ja.dbpedia.org/resource/霧島市	http://ja.dbpedia.org/property/ホ	ty...	26	673	3
http://ja.dbpedia.org/resource/鹿嶋市	http://ja.dbpedia.org/property/ホ	y/...	33	666	4
http://ja.dbpedia.org/resource/君津市	http://ja.dbpedia.org/property/ホ	/...	33	666	4
http://ja.dbpedia.org/resource/清須市	http://ja.dbpedia.org/property/ホ	id...	30	669	4
http://ja.dbpedia.org/resource/江田島市	http://ja.dbpedia.org/property/ホ	:d...	20	679	2
http://ja.dbpedia.org/resource/珠洲市	http://ja.dbpedia.org/property/ホ	'e...	24	675	3
http://ja.dbpedia.org/resource/小金井市	http://ja.dbpedia.org/property/ホ	ja...	31	668	4
http://ja.dbpedia.org/resource/那須烏山市	http://ja.dbpedia.org/property/ホ		84	615	12
http://ja.dbpedia.org/resource/稲敷市	http://ja.dbpedia.org/property/ホ	[illegible]	21	678	3

Figure 54 - Inner Profiling showing missing predicates for Japanese template

The highest completeness found was 31% for the resource “佐賀市” (Saga). The lowest percentage, however, was the same as in the Brazilian, 1% for the resource 行橋市 (Yukuhashi).

Although both templates kept their percentages below the average, the Japanese was slightly better. Besides having a higher completeness in one resource, when comparing the quantity of resources that had 23% (the highest percentage in the Brazilian counties) the amount in the Japanese version was 5, against 3 in DBpedia pt.

To sum up, although these datasets were not well qualified in this plugin, their percentages are understandable and even acceptable, because their resources may contain exclusive properties, which led to a low final completeness.

4.2 COVID-19 DATASETS

Taking into consideration the urgent need for information in a time where COVID-19 is the main topic for bringing such terrible amount of deaths, a study focusing on some official datasets containing its data was essential.

The two datasets used in this analysis were Open DataSUS[54], an official dataset provided by the Brazilian government, and Harvard Dataset[55], a repository built by Harvard University to store, among others, COVID-19's data from all around the world.

For a better approach concerning COVID-19 datasets, three tests were created (figure 55) in order to provide meaningful results on the data used.

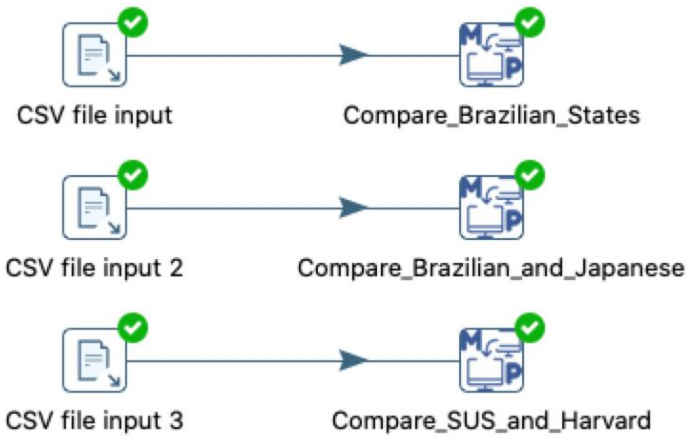


Figure 55 - Kettle flow to analyze COVID-19's data

First, all datasets had to go through some transformations before their usage in the Merge Profiling plugin, mostly because they had to be either in the Subject/predicate/value or N-triple format, to be used as input.

To achieve this, a simple Python program[56] got the CSV fields and one-by-one transformed them into predicate and object, using the "state" as subject.

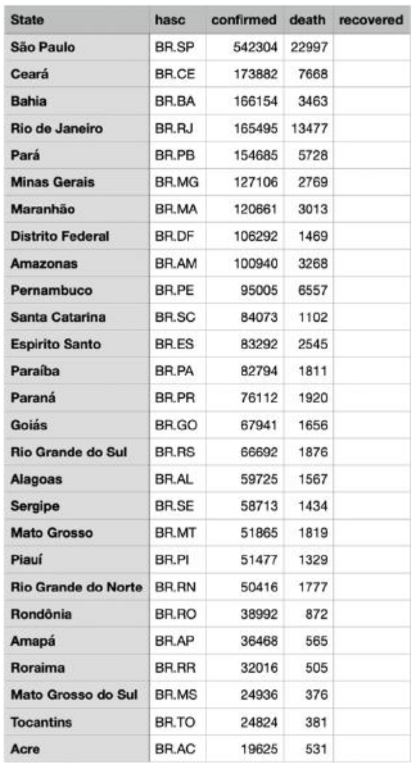
[54] https://opendatasus.saude.gov.br/dataset
[55] https://dataverse.harvard.edu/
[56] https://github.com/ingridpacheco/NormalizeDataset

4.2.1 Comparing Brazilian values

The first analysis was done inside the Brazilian dataset provided by Harvard[57] in two different days. To get a better overview of how the number of deaths and confirmed cases increased over time, the first data was taken from the beginning of COVID-19 in Brazil, on April 6th, while the second was more recent, taken from August 2.

They were downloaded having the properties "state", "hasc" (the state's acronyms), "confirmed" (total cases confirmed till the moment), "death" (total amount of deaths) and "recovered" (figure 56).

It is important to mention that while the field "recovered" was null inside the Brazilian states, it was a valid field with a reasonable number in plenty of other datasets.

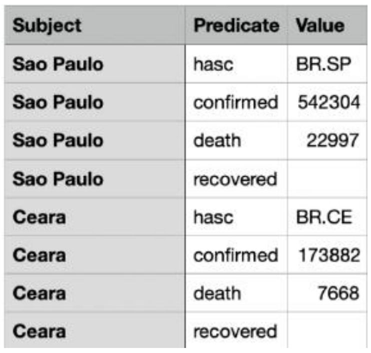


State	hasc	confirmed	death	recovered
São Paulo	BR.SP	542304	22997	
Ceará	BR.CE	173882	7668	
Bahia	BR.BA	166154	3463	
Rio de Janeiro	BR.RJ	165495	13477	
Pará	BR.PB	154685	5728	
Minas Gerais	BR.MG	127106	2769	
Maranhão	BR.MA	120661	3013	
Distrito Federal	BR.DF	106292	1469	
Amazonas	BR.AM	100940	3268	
Pernambuco	BR.PE	95005	6557	
Santa Catarina	BR.SC	84073	1102	
Espirito Santo	BR.ES	83292	2545	
Paraíba	BR.PA	82794	1811	
Paraná	BR.PR	76112	1920	
Goiás	BR.GO	67941	1656	
Rio Grande do Sul	BR.RS	66692	1876	
Alagoas	BR.AL	59725	1567	
Sergipe	BR.SE	58713	1434	
Mato Grosso	BR.MT	51865	1819	
Piauí	BR.PI	51477	1329	
Rio Grande do Norte	BR.RN	50416	1777	
Rondônia	BR.RO	38992	872	
Amapá	BR.AP	36468	565	
Roraima	BR.RR	32016	505	
Mato Grosso do Sul	BR.MS	24936	376	
Tocantins	BR.TO	24824	381	
Acre	BR.AC	19625	531	

Figure 56 - Original Harvard data

After the transformation all headers, besides "state", became a predicate with their values in the object field. Therefore, each state got 4 predicates: "hasc", "confirmed", "death" and "recovered" (figure 57).

---
[57] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/YB2S7D



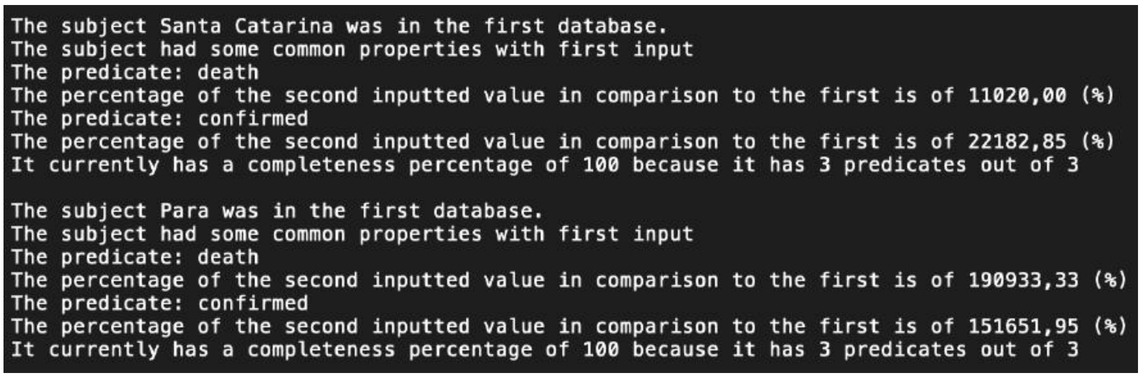
Subject	Predicate	Value
Sao Paulo	hasc	BR.SP
Sao Paulo	confirmed	542304
Sao Paulo	death	22997
Sao Paulo	recovered	
Ceara	hasc	BR.CE
Ceara	confirmed	173882
Ceara	death	7668
Ceara	recovered	

Figure 57 - Transformed data

Next, the "CSV file input" was called to get the CSV fields and use them as input for Merge Profiling. Then, the last received as input both the previous fields returned from the other plugin and the new CSV input with the data from the Brazilian states on April 6th.

The plugin got the subjects with their predicates and values and compared them to the ones found on the other input. Moreover, it is valuable to inform that the meta-model used for comparison in this test was Harvard's, meaning that is was expected for every dataset to contain valid values in all four predicates.

In addition, this specific test was focused on finding a correlation between different values in the same predicate, to understand how they increased/decreased over time (figure 58) following the mathematical expression: (100*newValue)/firstValue.



```
The subject Santa Catarina was in the first database.
The subject had some common properties with first input
The predicate: death
The percentage of the second inputted value in comparison to the first is of 11020,00 (%)
The predicate: confirmed
The percentage of the second inputted value in comparison to the first is of 22182,85 (%)
It currently has a completeness percentage of 100 because it has 3 predicates out of 3

The subject Para was in the first database.
The subject had some common properties with first input
The predicate: death
The percentage of the second inputted value in comparison to the first is of 190933,33 (%)
The predicate: confirmed
The percentage of the second inputted value in comparison to the first is of 151651,95 (%)
It currently has a completeness percentage of 100 because it has 3 predicates out of 3
```

Figure 58 - Merge Profiling result

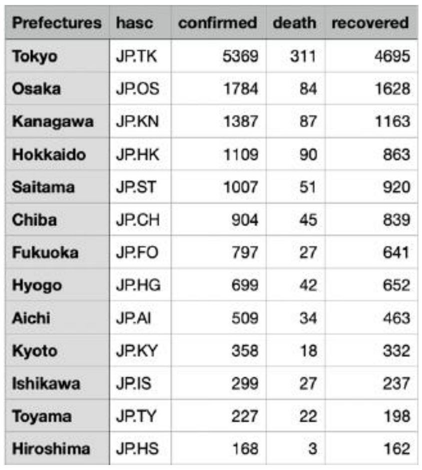
With the results, it was clear that the amount of deaths and quantity of confirmed cases increased absurdly within 4 months, from April to August. The state that had the highest increase in the amount of deaths was Pará, with an impressive percentage of 190.933%, when compared to the first quantity. Actually, it is totally understandable because when COVID-19 started there were only 3 deaths there, against 5728 now.

On the other hand, the state with the highest increase in confirmed cases was Rondônia, with 278.514,28%. On April it had 14 confirmed cases, while now it has 38992.

Lastly, besides Harvard dataset having a mistype in Para's acronym, writing PB instead of PA, it did not affect the result considering it was based on the state's full name, and not the acronym.

4.2.2 Comparing different countries

After the first test, it was time to move forward and analyze distinct datasets from the same repository, in this case, the Brazilian and Japanese[58] (figure 59) data from June 6th, both from the Harvard dataset.



Prefectures	hasc	confirmed	death	recovered
Tokyo	JP.TK	5369	311	4695
Osaka	JP.OS	1784	84	1628
Kanagawa	JP.KN	1387	87	1163
Hokkaido	JP.HK	1109	90	863
Saitama	JP.ST	1007	51	920
Chiba	JP.CH	904	45	839
Fukuoka	JP.FO	797	27	641
Hyogo	JP.HG	699	42	652
Aichi	JP.AI	509	34	463
Kyoto	JP.KY	358	18	332
Ishikawa	JP.IS	299	27	237
Toyama	JP.TY	227	22	198
Hiroshima	JP.HS	168	3	162

Figure 59 - Original Harvard Japan's data

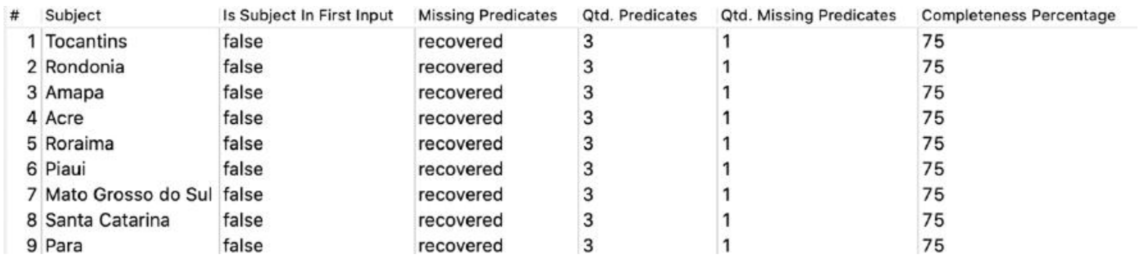
The Japanese dataset was chosen to continue the set of comparisons between Japanese and Portuguese/Brazilian versions.

It started just as the last time, transforming the CSV file and, later, calling the plugin Merge Profiling, aiming to compare these two datasets and find missing predicates.

Before even using the mentioned plugin, it was easy to realize that the Japanese version had a predicate that was not in the Brazilian, the recovered data. The most valuable lesson taken from this is that the Brazilian government (entity from which these data are coming) may not be releasing full access to open data, delivering them in pieces, while the Japanese government is providing them fully for future uses.

After the inputs were set, it ran and returned the expected data, showing that the predicate "recovered" was missing in the Brazilian version, or, at least, its value, leading to a 75% completeness for all states (figure 60).

[58] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/24EXUH



#	Subject	Is Subject In First Input	Missing Predicates	Qtd. Predicates	Qtd. Missing Predicates	Completeness Percentage
1	Tocantins	false	recovered	3	1	75
2	Rondonia	false	recovered	3	1	75
3	Amapa	false	recovered	3	1	75
4	Acre	false	recovered	3	1	75
5	Roraima	false	recovered	3	1	75
6	Piaui	false	recovered	3	1	75
7	Mato Grosso do Sul	false	recovered	3	1	75
8	Santa Catarina	false	recovered	3	1	75
9	Para	false	recovered	3	1	75

Figure 60 - Merge Profiling result for Brazilian dataset considering Japanese dataset

As the subjects were not the same (each was a different state), it did not compare values and only delivered the missing predicates and the states percentages.

4.2.3 Comparing different repositories

The final test was bolder and went out of the box, because Harvard's repository is not the only place where COVID-19 data can be found. Therefore, Open DataSUS, an open data repository provided by the Brazilian government, was used to allow this comparison.

When this use case was thought and executed, daily data related to COVID-19 cases were still available to download, while now only weekly data is on their website[59]. It is definitely worrying, as these data supported researches and are no longer available.

On the government's website, there is still a good amount of open data to download. However, the mentioned contained the deaths, confirmed cases and more information till that chosen day, while the accumulative data joins them by weeks.

Moreover, the Open DataSUS website is a little tricky when it comes to trying to find the correct dataset to download. It currently has 4 labels to COVID-19, and none of them refer to confirmed cases. This is only available on COVID-19's website, that shows graphs and statistics about these data, once again, accumulatively.

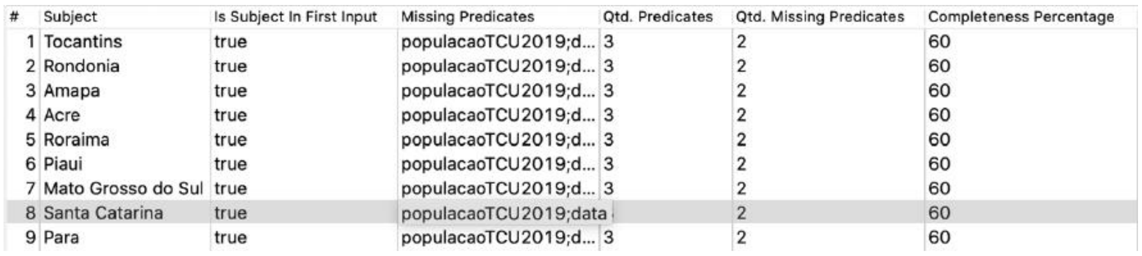
With the previously downloaded data[61] from June 6th, and also Harvard's data, it was time to transform them.

The government dataset suffered some modifications in its predicates names to match the predicates in Harvard's data, as they meant the same thing with different names. Furthermore, there were also some fields and rows that were deleted because they were not meaningful for this analysis.

[59] https://covid.saude.gov.br/

[61] https://github.com/ingridpacheco/ETL4Profiling/blob/master/usage/COVID-19/hoje_painel_covidbr_06jun2020.csv

Finally, with the modified data, the CSVs were used as input for Merge Profiling and the result showed the missing predicates in Harvard's data (figure 61).



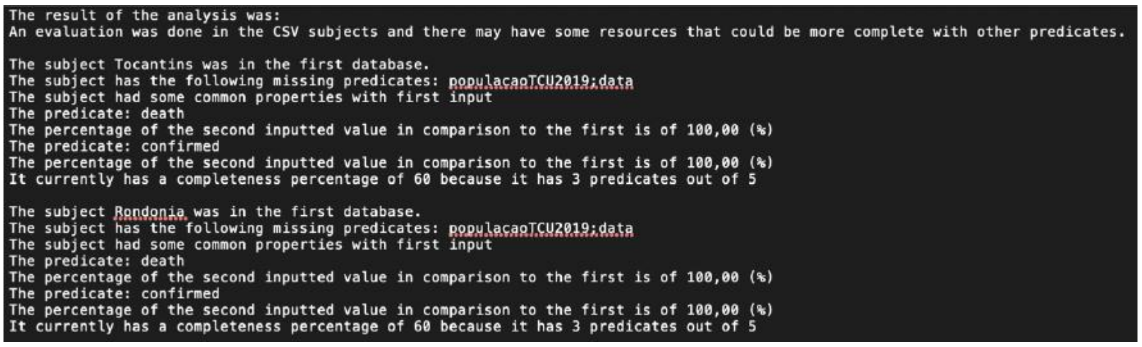
#	Subject	Is Subject In First Input	Missing Predicates	Qtd. Predicates	Qtd. Missing Predicates	Completeness Percentage
1	Tocantins	true	populacaoTCU2019;d...	3	2	60
2	Rondonia	true	populacaoTCU2019;d...	3	2	60
3	Amapa	true	populacaoTCU2019;d...	3	2	60
4	Acre	true	populacaoTCU2019;d...	3	2	60
5	Roraima	true	populacaoTCU2019;d...	3	2	60
6	Piaui	true	populacaoTCU2019;d...	3	2	60
7	Mato Grosso do Sul	true	populacaoTCU2019;d...	3	2	60
8	Santa Catarina	true	populacaoTCU2019;data		2	60
9	Para	true	populacaoTCU2019;d...	3	2	60

Figure 61 - Merge Profiling result of Harvard data considering Open DataSUS

This time the meta-model used as reference was the Open DataSUS dataset, as it was more complete and could point out improvement fields for Harvard's.

As the government data had more predicates, the completeness for Harvard's subjects was slightly above the average (60%), with two missing predicates that were only on Open DataSUS (populacaoTCU2019 and data).

Moreover, it shows that besides Harvard's data having a good quality, it could be better by receiving important fields that would help to get other statistics, for example, the percentage of deaths considering the population from that area.



```
The result of the analysis was:
An evaluation was done in the CSV subjects and there may have some resources that could be more complete with other predicates.

The subject Tocantins was in the first database.
The subject has the following missing predicates: populacaoTCU2019;data
The subject had some common properties with first input
The predicate: death
The percentage of the second inputted value in comparison to the first is of 100,00 (%)
The predicate: confirmed
The percentage of the second inputted value in comparison to the first is of 100,00 (%)
It currently has a completeness percentage of 60 because it has 3 predicates out of 5

The subject Rondonia was in the first database.
The subject has the following missing predicates: populacaoTCU2019;data
The subject had some common properties with first input
The predicate: death
The percentage of the second inputted value in comparison to the first is of 100,00 (%)
The predicate: confirmed
The percentage of the second inputted value in comparison to the first is of 100,00 (%)
It currently has a completeness percentage of 60 because it has 3 predicates out of 5
```

Figure 62 - Merge Profiling report file showing that values from both Open DataSUS and Harvard are the same

Another useful observation is that the datasets presented the same value when it comes to the same predicates, demonstrated by the 100% in the comparison (figure 62), which indicates that besides not being fully complete, Harvard's data are correct and correspond to official data released by the government, proving their legitimacy.

4.3 FINAL CONSIDERATIONS

Throughout the entire experiment plenty of information came out, which helped in the creation of a profile for each searched template and dataset. It was also possible to understand how metadata were mapped inside the resources and how complete they were.

Furthermore, much was discovered about the templates, such as their completeness in relation to both their properties and resources, and their particularities in terms of accuracy, helping to fully comprehend the result.

Even though the first tests were only focused in two templates, each in a different version, it was a good start for a deeper analysis. There is an abundance of other templates and DBpedia versions that were not explored yet but that have fascinating information waiting to be found, so as the COVID-19 datasets used in the final tests.

This is just the top of the iceberg, as much still remains to be discovered.

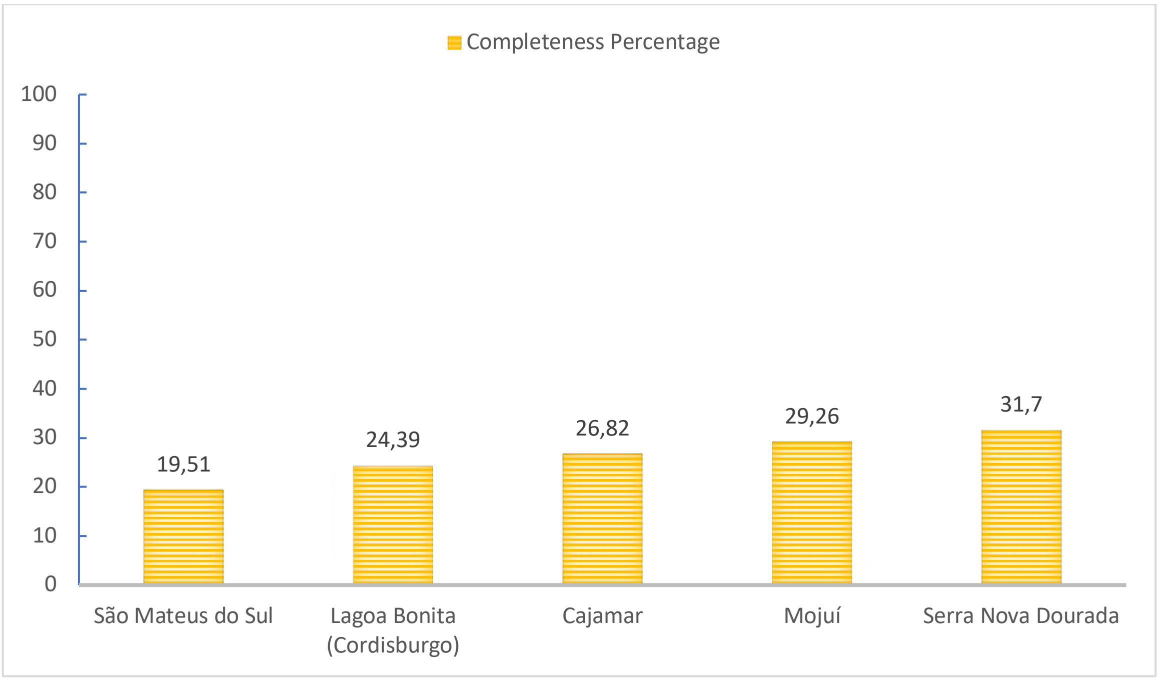
5 CONCLUSIONS

The main goal of this study was to give the initial step in dataset profiling, aiming to improve the semantic web universe, first taking DBpedia as a case study and then generalizing it.

Even though there are some tools that clean and transform data, as Open Refine and CEDAR, they focus on receiving massive datasets/metadata and processing/improving them, instead of evaluating completeness directly from DBpedia using its templates as meta-model. Therefore, this project was extremely helpful for the reuse of the repository, as now is easier to perceive its major issues to work on.

Despite the fact that it has an acceptable percentage, it still has some incomplete resources and templates that have to be fixed, which also served as motivation for this project. For example, according to Graph 1 it is possible to see that some resources have their percentages below the average, leading the template to a completeness of 58.3%.

Graph 1 - Worst Brazilian cities resources



Source: The author

As FAIR principles are based on mechanisms to facilitate openness, interlinking and reuse of data, a good completeness and accuracy help with this objective. Moreover, as it is written in the R1 (Reuse) page[70], it will be easier to find/reuse data if there are plenty of

[70] https://www.go-fair.org/fair-principles/r1-metadata-richly-described-plurality-accurate-relevant-attributes/

metadata attached to them, even if they are insignificant. Hence, it was easy to realize that the project should focus on data quality, evaluating DBpedia completeness in terms of their metadata.

An enormous part of this work was focused on how to analyze and generate a data profiling for any dataset, but especially DBpedia. The template properties enlightened a path to study, making it possible to create more coherent comparisons according to the chosen plugin.

For the most generic plugins the assessment was also about the subjects' predicates, both inside the same and different datasets. First, the intention was to find inconsistences inside them, for instance, a predicate that is missing in a subject but is present in another. Then, it was to compare values in different versions, to verify if they increased or decreased.

To finish, another important observation is that although the plugins were able to identify some major problems that already existed in DBpedia, they also needed to look for possible ways to contribute with newer data, leading to the creation of Resource Input Analyzer, that checks if the properties from a possible resource are complete considering the template it is going to belong, aiming to increase the dataset completeness.

To sum up, with ETL4Profiling the user can identify the attention points and correct them, while inserting correct and complete data into the database, ensuring their modifications will not decrease the current completeness and accuracy of the dataset. Besides, it also helps to bring more accurate relations between data, in order to benefit future projects that will use it.

5.1 DIFICULTIES FOUND

5.1.1 Decide the work course

The first difficulty found was deciding the right software to develop. As discussed in chapter 3, the original intention was to find DBpedia maturity level based on FAIR aspects[71], but it became complicated because the majority of the evaluation was qualitative, which means that what it asked to assess was based on a person's opinion, for example, to indicate which of four statements was most applicable to the dataset.

The idea, however, of helping DBpedia to reach FAIR standards was not entirely discarded. The second attempt was to create a query automator that would find all the resources

[71]https://docs.google.com/spreadsheets/d/1gvMfbw46oV1idztsr586aG6-teSn2cPWe_RJZG0U4Hg/edit#gid=2080819087

from a class, however, again, this did not work. It was meant to automatically create a query starting from something the person wrote, but it had to match perfectly a template or class, becoming a nonviable idea in that moment.

While there was still the intention of understanding how the resources were connected to classes and templates, an opportunity came up and brought clarity to the project. In short, it was to check if all Brazilian counties were described in DBpedia, and, for this, a SPARQL query was required.

To start, the initial idea was to find parameters that could be used to find counties, and so resources that had the type City and the word "município" in their descriptions were searched. Nevertheless, there were counties that did not have it, therefore changing the route of the investigation.

Through an analysis with the official list provided by IBGE, it was possible to realize that their resources on DBpedia had a pattern, the property "dbp:wikiPageUsesTemplate". That is when DBpedia mapping page[72] appeared.

When the program ran it found some of the Brazilian resources and compared them to the ones from IBGE. As a result it appeared as some of them supposedly were not on DBpedia, but after a manual check trying to find them by their names it was proved that they were only mistyped. Hence, the real problem appeared, resources had missing properties and typos that needed to be fixed.

Finally, the idea of doing a data profiling on DBpedia brought back the memory of increasing the amount of metadata to help with the repository's reuse. The improvement to any other dataset was just a progressive advance, and the plugins came into sight fitting perfectly the project proposal.

5.1.2 Pentaho and its plugins documentation

Soon after deciding what should be implemented, the technology also had to be chosen, and that is when Pentaho plugins emerged as a clarification.

Among other possibilities, plugins were the best decision because of their reuse in the ETL4LOD context. They could be implemented as a Python library, but that would not allow their usage along with other tools such as ETL4LOD+ and ETL4DBpedia, which is something desired, as described in the future works section.

[72] http://mappings.dbpedia.org/index.php/Mapping_pt

Therefore, with the decision of creating plugins, some time had to be taken in order to study their creation and understand how Pentaho works.

The biggest problem in all this process was to learn the required steps to implement them. While there is a paucity of documentation in the Pentaho website about the creation of new plugins[73], the ones available are very simple and only teach the basic steps. If a more complex design and implementation is expected, the specifications are not enough, which is the case of ETL4Profiling.

In the above-mentioned tutorial, they explain the definitions of the plugins' most used functions, but they do not explain how to use them properly. Moreover, the example they give only writes the same word in plenty of rows, not supporting any other process. Finally, they do not explain how to manipulate data inside the necessary files and neither how to input them in the application.

All this knowledge was received from other students that are and were working with the same software, creating plugins, and through attempts and failures, trying to find the best way to make plugins perform what they were intended to.

Another project that assisted in the development of the plugins was ETL4LOD+. It was created some time ago by João Curcio, aiming to evolve ETL4LOD tools to work with Linked Open Data. As it is an open project, all its code is available on GitHub[75], and served as inspiration to understand how to write new plugins and incorporate them on Pentaho.

5.1.3 Tests

The third and last difficulty of this project was testing the plugins.

First of all, to really test the plugin it was necessary to build a new version every time a change was done, and then run Spoon (Kettle interface) to see if any errors appeared, what was usual considering the lack of documentation the software has. It can be acceptable to occur once, but in a continuous timeline this ends up becoming a trammel, making it difficult to code in a flow.

Another point to be noticed is the extended time of these tests. Some of them, because of their long processing, took a prolonged time to return a result. In order to assuage this situation, CSV files identified by their templates were created storing resources with their

[73] https://help.pentaho.com/Documentation/8.2/Developer_Center/PDI/Extend/000

[75] https://github.com/johncurcio/ETL4LODPlus

respective properties and values. Therefore, whenever a plugin requested data from DBpedia it tried first to fetch the file and then the SPARQL endpoint, decreasing the processing duration.

Obviously, all the plugins have some improvements on the way, which will be discussed in the next section, however, for what they were supposed to deliver, the tests worked fine.

5.2 FUTURE WORK

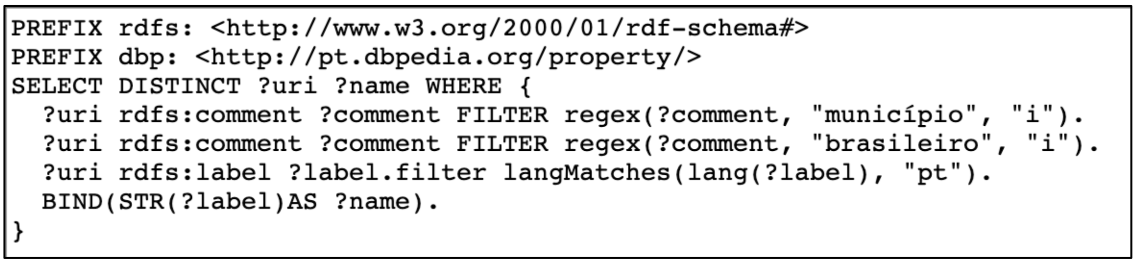
Each plugin has its own independence and functionality. Although they are fulfilling their purposes, there are some additional changes that would make them perform better and deliver more interesting data.

As a general change, it would be fascinating to broaden the focus of the study to analyze not only DBpedia templates but also its classes. Even though they are more generic in terms of classification of entities, for example, Brazilian cities and France cities are classified as "City", they tend to describe accurately the resources, and, consequently, would be a great query parameter.

Besides, it would also be flashy to create the query automator. The idea for this program would be how to find the exact data the user inputs, even considering typos.

The plugin would give the user the opportunity to decide whether they want an option loaded automatically or to put their own words, aiming to find resources/templates/properties that could match them.

It would open numerous possibilities for the plugins, such as the creation of a SPARQL query (figure 63) that would search the inputted word in the comment section of a resource or a Web scraper in the DBpedia mapping page that would find a template with the word in a substring of the template name.



```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://pt.dbpedia.org/property/>
SELECT DISTINCT ?uri ?name WHERE {
  ?uri rdfs:comment ?comment FILTER regex(?comment, "município", "i").
  ?uri rdfs:comment ?comment FILTER regex(?comment, "brasileiro", "i").
  ?uri rdfs:label ?label.filter langMatches(lang(?label), "pt").
  BIND(STR(?label)AS ?name).
}
```

Figure 63 - SPARQL query to get resources with specific words in the comment

Certainly, the openness of the text input could bring a lot of syntax problems and typos, that would make it harder to find the exact correspondence in DBpedia. For this, a string comparison could contribute to increase the accuracy percentage. For instance, above 80% the template name could be considered an option, letting the user decide among some of them the one they want to use.

In addition, there is another set of plugins, ETL4DBpedia, that could work complementarily with ETL4Profiling. As the plugins in this project were focused on finding the main flaws that already existed in diverse datasets, they could be used as an entry point for ETL4DBpedia, a project that inserts complete and accurate data in DBpedia PT. Finding the weaknesses that exist in the repository could guide the entrances they should receive. Accordingly, a helpful work would be their integration into a single consolidated data insertion project, that finds the biggest gaps and fixes them by inserting the desired data.

Later, it would be phenomenal to start analyzing datasets through other goals besides the ones explored in this study, such as to find subjects that do not have a specific predicate or with a value greater than a selected number.

Moreover, the expansion of comparisons would also be outstanding, for instance, comparing either different templates inside the same version or even all versions DBpedia has. These extensions could be easily implemented using the plugins as reference, taking into consideration the main core would be the same for them.

Lastly, ETL4Profiling was first projected focusing on completeness, one of the possible evaluations inside data quality. However, there are other assessments inside this field, as accuracy and consistency, that should be developed to improve the project and give its name a real meaning, providing a full data profiling for datasets.

The suggestions mentioned above are a more general belief of what could be incremented in the entire project, but each plugins also have its own enhancements.

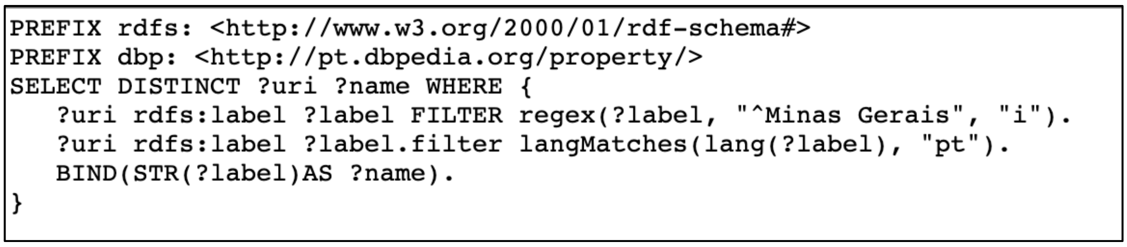
5.2.1 Template Property Analyzer and Template Resource Analyzer

All the properties have a type their values should correspond, and sometimes when resources instantiate them, they end up with the wrong type, for instance, “aniversário”, that should be a date but, in the resource “Minas Gerais”, is an integer. A good aspect to investigate would be how many properties have this issue so that an assessment of its accuracy could be possible.

Another viable change could be the output of the plugin. Instead of only writing CSV and TXT files, it could output a graph comparing the results for different entries, as illustrated at the beginning of this chapter. A graph, sometimes, is the best way to analyze an outcome, so it would be satisfying to return it as a result.

5.2.2 Resource Properties Analyzer

Some properties have literal values, like birthdate or nickname, but others could have resources related to them, for example, State. They currently have two possible values, a literal (object) or a resource (subject). If the value is a literal, the program could run a SPARQL query (figure 64) to verify if it has a related resource (the resource exists but in the property it is a literal instead of another subject). If it does, the program gives the user the chance to switch the value of the property to the resource URI.



```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://pt.dbpedia.org/property/>
SELECT DISTINCT ?uri ?name WHERE {
   ?uri rdfs:label ?label FILTER regex(?label, "^Minas Gerais", "i").
   ?uri rdfs:label ?label.filter langMatches(lang(?label), "pt").
   BIND(STR(?label)AS ?name).
}
```

Figure 64 - SPARQL query searching for a resource with the literal value found in "estado"

5.2.3 Property Analyzer

When it comes to the Property Analyzer, a possible improvement could be besides the delivery of the current information, the opportunity to create a value for each property if they do not exist in a certain resource.

While it is not easy to change a resource property value, mostly because it comes straight from Wikipedia, it is changing due to ETL4DBpedia. Now, adding data to DBpedia is becoming possible through direct publication on Wikipedia (NGOMO, Jean Gabriel N., 2020), which will certainly assist to fix missing properties.

5.2.4 Template Resource Input Analyzer

The Template Resource Input Analyzer is one of the most important plugins created, mainly because of the possibility to find missing resources in a template.

A valuable enhancement that could be implemented is the string comparison mentioned before. As the input is a CSV file, some level of accuracy has to be accepted, considering the high probability of containing incorrect data or typos.

Lastly, another improvement could be with the output. It would be a great adjustment if, instead of only returning the missing resources, it also created a file or RDF graph for each of them, along with the template properties and a default value, that then could be used as input either for Resource Input Analyzer or ETL4DBpedia.

5.2.5 Resource Input Analyzer

First, it would be interesting to add the missing properties with a default value in the selected resource, instead of only warning about them. It could also give the user the chance to change their values to make them more correct.

Another potential adjustment would be the insertion of the input into the respective template it belongs. While now the plugin does not have a direct access to DBpedia, which means that it cannot insert or modify its data, it could change in the future, uploading complete resources into the repository.

5.2.6 Inner Profiling and Merge Profiling

Finally, the Inner Profiling and Merge Profiling plugins are receiving the datasets considering they are already triplified or in the subject/predicate/object format. An improvement could be the specification of the CSV fields, because currently they consider the first column (if N-triple) or the first three columns (subject/predicate/object) as being the ones they should evaluate, not taking into consideration that maybe there are more fields in this CSV. This change would contribute enormously to the assessment of the dataset.

5.2.7 Maintenance

As the software is the initial state for dataset profiling, some upgrades will be necessary as time goes by, either to fix some bugs, update the documentation, improve the performance or to refactor plugins as DBpedia changes come. For this reason, as it is an extensive job, the software is open source and is available on GitHub[77], with documentation to help people that may want to work on it.

[77] https://github.com/ingridpacheco/ETL4Profiling

REFERENCES

AUER, Sören et al. Dbpedia: A nucleus for a Web of open data. **In: The semantic Web**. Springer, Berlin, Heidelberg, 2007. p. 722-735.

BERNERS-LEE, T.; HENDLER, J. e LASSILA, O. The Semantic Web. **Scientific American**, v.284, n.5, p. 29-37. 2001.

BERNERS-LEE, T. **Linked Data**. 2009. Available from: <https://www.w3.org/DesignIssues/LinkedData.html>. Retrieved: Feb. 23, 2020.

BIZER, Christian; HEATH, Tom; BERNERS-LEE, Tim. Linked data: The story so far. In: **Semantic services, interoperability and Web applications: emerging concepts**. IGI Global, 2011. p. 205-227.

BOULTON, Geoffrey et al. The Royal Society. **Science as an open enterprise**. London, 2012.

BRANDUSESCU, Ana; IGLESIAS, Carlos. World Wide Web Foundation. **Open Data Barometer - Leaders Edition**, Washington DC: World Wide Web Foundation, 2018. Available from:<https://opendatabarometer.org/doc/leadersEdition/ODB-leadersEdition-Report.pdf>. Retrieved: Sep. 17, 2019.

CASTERS, Matt; BOUMAN, Roland; VAN DONGEN, Jos. **Pentaho Kettle solutions: building open source ETL solutions with Pentaho Data Integration**. John Wiley & Sons, 2010.

COLLINS, Sandra et al. **Turning FAIR into reality: Final report and action plan from the European Commission expert group on FAIR data**. 2018.

CORDEIRO, et al. **An approach for managing and semantically enriching the publication of Linked Open Governmental Data. In: Workshop de Computação Aplicada em Governo Eletrônico (WCGE)**, 2011, Florianópolis. Workshop de Computação Aplicada em Governo Eletrônico (WCGE), 2011. v. 1.

DOAN, AnHai; HALEVY, Alon; IVES, Zachary. **Principles of data integration**. Elsevier, 2012.

FÄRBER, Michael et al. A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago. **Semantic Web Journal**, v. 1, n. 1, p. 1-5, 2015.

FASTER, Data Sources. **Pentaho Data Integration**. 2014.

ALLEY, Garrett. **What is Data Profiling?**. 2019. Available from:< https://www.alooma.com/blog/what-is-data-profiling> Retrieved: Oct. 17, 2019.

GO FAIR. **What is an Implementation Network?**. Leiden, 2017. Available from: <https://www.go-fair.org/implementation-networks/> Retrieved: March 11, 2020.

HEBELER, John et al. **Semantic Web Programming**. Oreilly & Associates Inc. 2009.

HENDLER, James; BRNERS-LEE, T.; MILLER, Eric. **Integrating applications on the semantic web**. JOURNAL-INSTITUTE OF ELECTRICAL ENGINEERS OF JAPAN, v. 122, n. 10, p. 676-680, 2002.

HODSON, S., et al. **Turning FAIR Data into Reality. Interim report of the European Commission Expert Group on FAIR data**. 2018

INOUE, Hiroyuki; AMAGASA, Toshiyuki; KITAGAWA, Hiroyuki. An ETL framework for online analytical processing of linked open data. In: **International Conference on Web-Age Information Management**. Springer, Berlin, Heidelberg, 2013. p. 111-117.

ISMAIL, Salih; SHAIKH, Talal. ALiterature **REVIEW ON SEMANTIC WEB–UNDERSTANDING THE PIONEERS'PERSPECTIVE**. In: The Sixth International Conference on Computer Science, Engineering and Applications. 2016. p. 15-28.

JACOB, Elin K. **Ontologies and the semantic web**. Bulletin of the American Society for Information Science and Technology, v. 29, n. 4, p. 19-19, 2003.

JACOBSEN, Annika et al. A generic workflow for the data FAIRification process. **Data Intelligence**, v. 2, n. 1-2, p. 56-65, 2020.

LEHMANN, Jens et al. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. **Semantic Web**, v. 6, n. 2, p. 167-195, 2015.

MENDES, Pablo N.; JAKOB, Max; BIZER, Christian. **DBpedia: A Multilingual Cross-domain Knowledge Base**. In: LREC. 2012. p. 1813-1817.

MORSEY, Mohamed et al. **Dbpedia and the live extraction of structured data from wikipedia**. Program, 2012.

NAUMANN, Felix. **Data profiling revisited**. ACM SIGMOD Record, v. 42, n. 4, p. 40-49, 2014.

NGOMO, Jean Gabriel N. **UMA ABORDAGEM PARA MELHORIA DA QUALIDADE DA DBPEDIA PT COMO HUB DE DADOS DE PESQUISA**. 2020.

ROLDÁN, María Carina. **Pentaho 3.2 Data Integration: Beginner's Guide**. Packt Publishing Ltd, 2010.

SILVA, João Felipe Curcio da. **ETL4LOD+: evolução do suporte ao ciclo de publicação de dados conectados**. 2018.

STEFANI**,** Giovani. Pentaho, Tutoriais. **Pentaho Data Integration Step by Step – parte 01**, 2018. Available from:< https://www.infointelligence.com.br/2018/09/25/pentaho-data-integration-step-by-step-parte-01/
>. Retrieved: Nov. 23, 2019.
STERNE**,** Jonathan. **Plug-in**. 2019. Available from:<https://www.britannica.com/technology/plug-in>. Retrieved: Dec. 7, 2019.

TAYE, Mohammad Mustafa. **Understanding semantic web and ontologies: Theory and applications**. arXiv preprint arXiv:1006.4567, 2010.

VAZ, José Carlos; RIBEIRO, Manuella Maia; MATHEUS, Ricardo. **Dados governamentais abertos e seus impactos sobre os conceitos e práticas de transparência no Brasil**. Cadernos ppg-au/ufba, v. 9, n. 1, 2010.

W3C**. OWL.** 2012. Available from:<https://www.w3.org/OWL/>. Retrieved: Nov. 23, 2019.

W3C**. RDF.** 2014. Available from:<https://www.w3.org/RDF/>. Retrieved: Dec. 12, 2019.

W3C**. Semantic Web – W3C.** [2015?]. Available from:<https://www.w3.org/standards/semanticweb/>. Retrieved: Nov. 9, 2019.

WILKINSON, M.; DUMONTIER, M.; AALBERSBERG, I. *et al.* **The FAIR Guiding Principles for scientific data management and stewardship**. *Sci Data* **3,** 160018, 2016. Available from:< https://www.nature.com/articles/sdata201618#citeas
>. Retrieved: Sep. 21, 2019.

WORLD WIDE WEB CONSORTIUM et al. **RDF 1.1 concepts and abstract syntax**. 2014.

______. FAIR Principles. Available from:<https://www.go-fair.org/fair-principles/>. Retrieved: Oct. 7, 2019.

______. Fiocruz e IBICT marcam presença na Rede Internacional de Implementação do GO FAIR, 2019. Available from:< https://www.icict.fiocruz.br/content/fiocruz-e-ibict-marcam-presen%C3%A7a-na-rede-internacional-de-implementa%C3%A7%C3%A3o-do-go-fair>. Retrieved: Oct. 23, 2019.

APPENDIX A – WEB SCRAPER IN DBPEDIA MAPPING PAGE

```
public String[] getTemplateValues(String DBpedia){
    try {
        String url = String.format("http://mappings.dbpedia.org/index.php/Mapping_%s", DBpedia.toLowerCase());
        Document doc = Jsoup.connect(url).get();
        Elements mappings = doc.select(String.format("a[href^=\"/index.php/Mapping_%s:\"]", DBpedia.toLowerCase()));

        TemplateValues = new String[mappings.size()];
        for (int i = 0; i < mappings.size(); i++) {
            String templateMapping = mappings.get(i).text();
            TemplateValues[i] = templateMapping.split(":")[1];
        }

        return TemplateValues;
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        TemplateValues = new String[1];
        TemplateValues[0] = "";
        return TemplateValues;
    }
}
```

APPENDIX B – WEB SCRAPER IN RESOURCE PAGE.

```
public void getResourceProperties(String resource) {
    String DBpedia = meta.getDBpedia();
    String regexpValue = getMapValues(DBpedia);
    Set<String> resourcesProperties = new HashSet<String>();
    Integer counter = 0;

    try {
      String url = this.urls.getResourceUrl(resource.replace(" ", "_"));
      Document doc = Jsoup.connect(url).get();
      Elements properties = doc.select(String.format("a[href^=\"http://%s.dbpedia.org/property\"]", DBpedia));
      Elements values = doc.select(regexpValue);

      for (int i = 0; i < properties.size(); i++) {
        String resourceProperty;
        if (DBpedia.equals("ja") && values.get(i).toString().matches("^(a).*$")) {
          String actualProperty = values.get(i).attributes().get("rel").split("prop-ja:")[1];
          if (resourcesProperties.contains(actualProperty)) {
            resourceProperty = actualProperty;
          }
          else {
            resourceProperty = properties.get(counter).text().split(":")[1];
            resourcesProperties.add(resourceProperty);
            counter += 1;
          }
        }
        else {
          resourceProperty = properties.get(counter).text().split(":")[1];
          resourcesProperties.add(resourceProperty);
          counter += 1;
        }
        if (!data.resourceProperties.contains(resourceProperty)) {
          String propertyValue = values.get(i).text();
          getFormatedValue(resource, resourceProperty, propertyValue);
          if (meta.getOption() == "Template resources properties")
            cacheProperty(resourceProperty);
        }
      }
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
}
```