Assignment 1 – Animal Scouter

University of Aveiro

Web Semântica

2022/2023

Group nº 2:

- João Bernardo Coelho Leite - 115041
- João Pedro dos Reis - 115513
- Luís Miguel Gomes Batista - 115279

Index

Introduction

In current times, websites serve as powerful platforms for information dissemination, interaction, and engagement. As technologies continue to evolve, web developers are constantly seeking innovative ways to enhance user experiences and improve the functionality of their websites. One such avenue of exploration involves leveraging the capabilities of inferences and SPIN (SPARQL Inferencing Notation) to extract meaningful insights from data and enable advanced functionalities.

This report explores the potential of incorporating inferences and SPIN (SPARQL Inferencing Notation) into a website. It focuses on creating a new tab on an existing website, using a transformed dataset of animal features in N3 format OWL. The ontology editor Protégé is utilized to define rules, axioms, and constraints for managing inferences. Additionally, the integration of data from Wikidata and DBpedia is considered to enrich the dataset and enhance the website's functionality.

We aim to demonstrate the benefits of inferences and SPIN in improving the website's user experience and providing comprehensive information. By integrating external knowledge bases, the website can tap into a broader range of structured data, establish semantic connections, and offer a more interconnected web experience.

In summary, this paper showcases the possibilities and advantages of incorporating ontologies, inferences and SPIN into website design, as well as the gathering of external information from endpoint sources for additional content.

Data Operations (SPARQL)

Transformations

The initial dataset, represented in N-Triples format, contained statements in the form of subject-predicate-object triples, such as:

```
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/name> "Abelha" .
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/class> <http://zoo.org/class/id/6> .
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/legs> "6" .
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/nurt> <http://zoo.org/nurt/id/1> .
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/has> "Hair" .
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/is> "Venomous" .
<http://zoo.org/animal/id/abelha> <http://zoo.org/pred/is> "Airborne" .
```

Figure 1 Example of original dataset N-Triples formatting

To maximize the readability and flexibility of the dataset, a transformation process was undertaken to convert the dataset into N3 format using a python script converter [1]. This transformation enhanced the dataset's expressiveness and provided a more structured representation of the information.

The transformed dataset, now in N3 format, follows a different syntax that offers improved readability and organization. Each entity in the dataset is assigned a prefixed name to enhance clarity and facilitate efficient referencing.

In the N3 representation, the subject <http://zoo.org/animal/id/abelha> has been transformed into the prefixed name ‘zooi:abelha’, improving readability and providing a more concise identification. Similarly, the predicates and objects have been transformed into prefixed names and values, respectively, to ensure clarity and maintain consistency throughout the dataset.

```
zooi:abelha zoop:class zooc:6 ;
    zoop:has "Hair" ;
    zoop:is "Airborne",
        "Venomous" ;
    zoop:legs "6" ;
    zoop:name "Abelha" ;
    zoop:nurt zoon:1 .
```

Figure 2 Example of N3 formatted entry of the dataset

By employing OWL techniques and transforming the dataset into N3 format, we have enhanced the expressiveness and readability of the dataset. This transformation sets the foundation for more sophisticated reasoning and enables the utilization of advanced inference techniques and SPIN.

In addition, the attributes “Backbone” and “Predator” were added to the dataset to increase the possibility and accuracy of inferences in the future.

Ontology definition

Ontology (RDFS & OWL)

The ontology presented in this report defines a comprehensive taxonomy and relationships within the domain of animals. It serves as a structured knowledge representation that enables reasoning and inference about different types of animals, their attributes, and classifications.

The ontology is built using the RDF/OWL syntax, mainly created with the aid of Protégé. It begins by establishing the necessary prefixes for namespaces used in the ontology, such as 'owl', 'rdf', 'rdfs', and custom prefixes like 'zooc', 'zooi', 'zoon', and 'zoop', as can be seen in Figure 3.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix zooc: <http://zoo.org/class/id/> .
@prefix zooi: <http://zoo.org/animal/id/> .
@prefix zoon: <http://zoo.org/nurt/id/> .
@prefix zoop: <http://zoo.org/pred/> .
@prefix animalc: <http://zoo.org/class/> .
@base <http://www.w3.org/2002/07/owl#> .
```

Figure 3 Ontology Prefixes

It defines a hierarchy of classes that represent different types of animals. Classes such as 'animalc:Amphibian', 'animalc:Bird', 'animalc:Fish', 'animalc:Mammal', and 'animalc:Reptile' provide a structured taxonomy of animals based on their biological characteristics and behaviours. It also incorporates logical constructs, such as equivalent classes and restrictions, to capture relationships and constraints among classes.

By using these constructs, the ontology ensures consistency and enables inferencing. For example, 'animalc:Land' in the ontology, representing land-dwelling animals states that this is equivalent to the intersection of two classes. One is defined as the complement of animals that have the property 'zoop:is' with the value "Airborne", which excludes airborne animals. The other is defined as the complement of animals that have the same property with the value "Aquatic", excluding aquatic animals. This definition ensures that 'animalc:Land' includes only animals that are neither airborne nor aquatic, implying they live on land.

```
animalc:Land rdf:type owl:Class ;
             owl:equivalentClass [ owl:intersectionOf ( [ rdf:type owl:Class ;
                                                          owl:complementOf [ rdf:type owl:Restriction ;
                                                                             owl:onProperty zoop:is ;
                                                                             owl:hasValue "Airborne"
                                                                           ]
                                                        ]
                                                        [ rdf:type owl:Class ;
                                                          owl:complementOf [ rdf:type owl:Restriction ;
                                                                             owl:onProperty zoop:is ;
                                                                             owl:hasValue "Aquatic"
                                                                           ]
                                                        ]
                                                      ) ;
                                        rdf:type owl:Class
                                      ] .
```

Figure 4 animalc:Land construct

Reasoner and SPIN Inferences

In this project, we used RDF and SPIN (SPARQL Inference Notation) to create a knowledge base of animal classifications. Through seven queries, we made significant inferences about different aspects of the animal kingdom. We identified "Arthropod animals" by classifying certain animals as arthropods based on their membership in the "Insect" class.

Regarding thermoregulation, we categorized animals as "Warm-Blood" if they had "Feathers" or "Hair," a "Backbone," and belonged to a specific nurturing class. Similarly, "Cold-Blood" animals were identified based on the presence of "Fins" or their classification as "Reptile," "Amphibian," and a specific nurturing class.

```
-- Warm-Blood animals
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX animalc: <http://zoo.org/class/>
PREFIX zoop: <http://zoo.org/pred/>
CONSTRUCT {
   ?s a animalc:Warm-Blood.
}
WHERE {
    {
        ?s a animalc:Animal.
        ?s zoop:has "Feathers".
    }
    UNION
    {
        OPTIONAL {
            ?s zoop:has "Hair".
        }
        ?s zoop:nurt zoon:2.
        ?s zoop:has "Backbone".
    }
}
```

Figure 5 Warm-Blooded animals query

We also created the inference of "Backbone animals" by classifying animals with "Feathers" or "Hair," a "Backbone," or "Fins," and belonging to relevant classes. Furthermore, we inferred "Land animals" by identifying animals in the "Animal" class without "Airborne" or "Aquatic" properties, categorizing them as land-dwelling. The inference of "No-Backbone animals" was created by identifying animals without a "Backbone," denoting those lacking a backbone. Lastly, as the reasoner could not infer the mammal class, an extra SPIN query was created to ensure this group of entities was referred to.

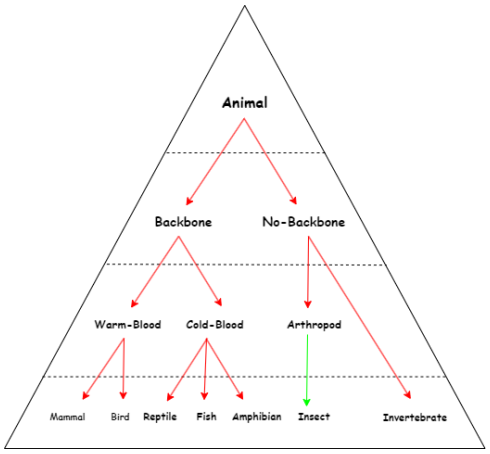

Figure 6 Animal class hierarchy (the green arrow shows that insect is a subclass which doesn't represent all of the Arthropods)

Use of Wikidata & DBpedia

The integration of Wikidata and DBpedia has greatly enhanced the functionality and information presented on our website. Through our connection to DBpedia, we were able to enrich the data presented in the 'Inferences' tab by incorporating additional details and context from DBpedia's extensive knowledge base. We present a brief description of the selected class/subclass and additional links for further investigation by the user. This allows us to provide users with a more comprehensive understanding of the inferred classifications and characteristics of various animals.

```
# dbpedia lookup
dbpedia_dict = {
    'Animal': 'Animal',
    'Backbone': 'Bone',
    'No-Backbone': 'Bone',
    'Warm-Blood': 'Lymph',
    'Cold-Blood': 'Lymph',
    'Arthropod': 'Species',
    'Mammal': 'Mammal',
    'Bird': 'Bird',
    'Reptile': 'Reptile',
    'Fish': 'Fish',
    'Amphibian': 'Amphibian',
    'Insect': 'Insect',
    'Invertebrate': 'Species'
}

query = """
        select ?info
        where {
            ?animal rdfs:label "_animal_type"@en.
            ?animal rdfs:comment ?info.
            filter(lang(?info)='en').
        }
        """
query = query.replace('_animal_type', dbpedia_dict[inference_entity])
sparqlDBpedia.setQuery(query)
results = sparqlDBpedia.query()
```

Figure 7 DBpedia query

Additionally, our query to Wikidata is used to retrieve images. By displaying images alongside the textual information, we enhance the visual appeal of the website and provide users with a more engaging and immersive experience.

```
# wikidata lookup
sparqlWikidata.setQuery("""
                        PREFIX wd: <http://www.wikidata.org/entity/>
                        PREFIX wdt: <http://www.wikidata.org/prop/direct/>
                        SELECT ?item ?itemLabel
                        WHERE
                        {
                            wd:Q729 wdt:P2716 ?item .
                            SERVICE wikibase:label {
                                bd:serviceParam
                                wikibase:language "en".
                            }
                        }
                        """)
# wdt:P2716 -> diversity image
# wdt:P18   -> bear image
wikidata_info = ''
for result in sparqlWikidata.query().bindings:
    #print('%s: %s' % (result["item"].value, result["itemLabel"].value))
    wikidata_info = result["itemLabel"].value
```

Figure 8 Wikidata query

Publication of semantics using Microformats

In order to demonstrate the capabilities of Microformats, we implemented these in the queries page. Whenever animals are “scouted” (selected) the loaded page will contain a brief hidden description of some of the attributes of each animal as well as a list of the “scouted” animals. This information can be validated through, for example, an online parser [2].


```
{
    "items": [
        {
            "type": [
                "h-card"
            ],
            "properties": {
                "scoutedanimals": [
                    "Anchovy",
                    "Baiacu",
                    "Barracuda",
                    "Cascudinho-De-Caverna",
                    "Flounder",
                    "Halibut",
                    "Lambari",
                    "Mackerel",
                    "Marlin",
                    "Matrinxa",
                    "Pirarucu",
                    "Raia-Chita",
                    "Tambaqui",
                    "Trout",
                    "Tubarao-Raposa"
                ]
            },
            "lang": "en"
        },
        {
            "type": [
                "h-card"
            ],
            "properties": {
                "name": [
                    "Marlin"
                ],
                "class": [
                    "Fish"
                ],
                "nurt": [
                    "Eggs"
                ],
                "has": [
                    "Fins",
                    "Backbone",
                    "Tail"
                ],
                "is": [
                    "Aquatic"
                ]
            },
            "lang": "en"
        }
    ]
```


Figure 9 Microformat parsed data of Cascudinho-de-Caverna

Application Features

The new tab incorporates three interactive buttons which will modify the dataset through inferences adding the attributes as “Land”, “No-Backbone” and “Mammal” which will affect the information shown on the rest of the website.

Additionally, it includes individual buttons organized in the same way as the class hierarchy previously defined. Clicking on any of these buttons will extend the page showing a brief description of the selected class fetched from DBpedia, a list of inferred animals from the dataset which are part of that class and links for the user to be able to further explore that specific class of animals and learn more on their own leisure.

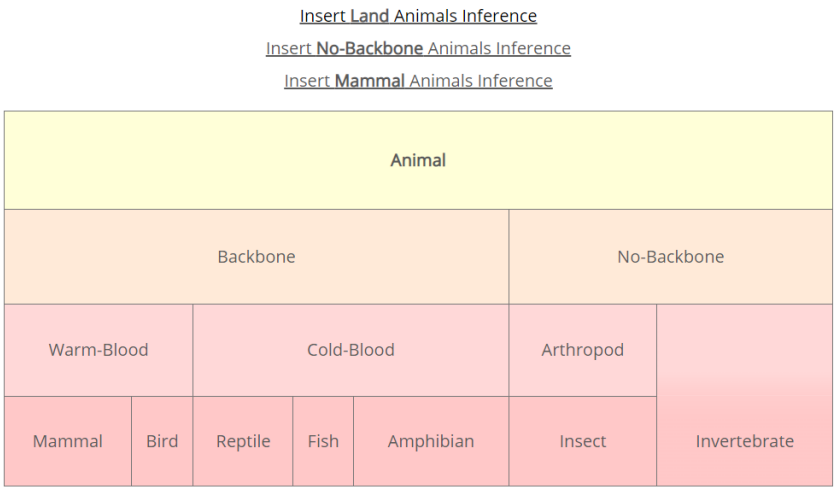

Figure 10 Inferences' Inserts and Class selection interactive buttons

We also added a picture gathered from Wikidata showing a few animals to enhance user experience.

Figure 11 Picture from Wikidata showcased in the inferences tab

Conclusions

The development of the new tab for our website has proven to be a resounding success. By incorporating inferences and leveraging data from external sources such as Wikidata and DBpedia, we have significantly enhanced the information presented to our users.

One of the key benefits of this new implementation is the utilization of SPIN, which has notably improved the speed of our queries. This enhancement has resulted in a slightly better user experience, allowing for faster retrieval and presentation of relevant data. While the impact may not be immediately noticeable due to the relatively small dataset we currently possess, it holds great potential for even greater efficiency and performance gains when applied to larger datasets.

By inferring additional knowledge based on existing data, we have been able to provide users with a more comprehensive understanding of the subject matter. This has enabled us to present a wider range of information, ensuring that our users can delve deeper into the topics they explore on our website.

In conclusion, the development of the new tab and implementation of inferences has been a success. Through the incorporation of inferences, integration of external data sources, and the utilization of SPIN, we have created a more informative and engaging experience for the users. While the immediate impact may be more pronounced with larger datasets, the foundations established lay the groundwork for even greater advancements in the future.

Application Configuration

Requirements

Installation requirements to set the developed application up and running:

- Python (preferably 3.8.10 or higher)
- GraphDB
- s4api (pip install s4api)
- sparqlwrapper (pip install sparqlwrapper)

Creating the database

In GrapDB control panel, it is needed to create a database named “zoo” with ruleset OWL2-RL and import the provided “zooall.n3” N3 file.

Optionally, for the base url, *http://zoo.org/* may be used.

Running with PyCharm

To run the application with PyCharm, simply open the wsproject folder and press the run button. Then, a localhost link should appear in the console which needs to be opened with a web browser.

Running with command line

For running the application using the command line, open a new command line in the “/wsproject/” directory and type the command “py manage.py runserver”. A localhost link should appear in the console which needs to be opened with a web browser.

References

Dataset

https://www.kaggle.com/datasets/agajorte/zoo-animals-extended-dataset

Slides

Representação do Conhecimento, Standards da Web Semântica, WS, DETI, UA

Representação do Conhecimento, A Linguagem SPARQL, WS, DETI, UA

Representação do Conhecimento, A *Triplestore* GraphDB, WS, DETI, UA

OWL – Web Ontology Language, WS, DETI, UA

A Inferência na Web Semântica, WS, DETI, UA

Dados Semânticos: Fontes de Dados, WS, DETI, UA

Publicação de Dados Semânticos, WS, DETI, UA

Web references

https://docs.djangoproject.com/en/4.1/ref/forms/widgets/

https://docs.djangoproject.com/en/4.1/ref/forms/fields/

[1] https://github.com/WimPessemier/rdfconvert

http://mappings.dbpedia.org/server/ontology/classes/

https://www.wikidata.org/wiki/Q729

[2] http://pin13.net/mf2/