DAO Ontology

Juliao Braga[1], Francisco Regateiro[2], Itana Stiubiner[1], Juliana Cristina Braga[1]

[1]Center for Mathematics, Computation and Cognition – Universidade Federal do ABC
Santo André – SP – Brazil

[2]Instituto Superior Tecnico – University of Lisbon
Lisbon – PT

j@ghaia.pt,f@ghaia.pt,{itana.stiubiener,juliana.braga}@ufabc.edu.br

Abstract. *This work presents a manually built ontology to aggregate knowledge of Decentralized Autonomous Organizations (DAOs) obtained from web pages. An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships between them, providing a common vocabulary for researchers to share information. Ontology construction from text involves analyzing collected text, identifying relevant terms and concepts, and representing the ontology using representation languages such as OWL, RDF, or RDFS. Manual ontology construction can be performed using applications such as Protégé. This work describes the methodology used, how to use the ontology created through Protégé using SPARQL, and presents future work proposals, including creating the same ontology using Deep Learning.*

Resumo. *Este trabalho apresenta uma ontologia construída manualmente para agregar conhecimento sobre Organizações Autônomas Descentralizadas (DAOs) obtidas a partir de páginas da web. Uma ontologia é uma descrição formal do conhecimento como um conjunto de conceitos dentro de um domínio e as relações entre eles, fornecendo um vocabulário comum para os pesquisadores compartilharem informações. A construção de ontologias a partir de texto envolve a análise do texto coletado, identificando termos e conceitos relevantes e representando a ontologia usando linguagens de representação como OWL, RDF ou RDFS. A construção manual de ontologias pode ser realizada usando aplicativos como o Protégé. Este trabalho descreve a metodologia utilizada, como usar a ontologia criada através do Protégé usando SPARQL e apresenta propostas de trabalhos futuros, incluindo a criação da mesma ontologia usando Deep Learning.*

1. Introduction

An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them. It ensures a common understanding of information and makes explicit domain assumptions, thus allowing organizations to make better sense of their data. By providing a common vocabulary for researchers who need to share information in a domain, it can help to share common understanding of the structure of information among people or software agents [Mishra and Sarika 2014] [Gruber 1993].

Ontology construction from text involves analyzing collected text for a specific domain of human knowledge, identifying relevant terms, concepts, and their relationships,

and representing the ontology using representation languages such as OWL (Web Ontology Language), RDF (Resource Description Framework), or RDFS (Resource Description Framework Schema) [Al-Aswadi et al. 2020]. Building domain-specific ontologies is a time-consuming and expensive manual construction task [Bangyal et al. 2022, Kietz et al. 2000].

Indeed, ontology construction is a complex process that involves analyzing a specific domain, identifying relevant concepts and relationships, and representing them in a formal language. Manual construction of ontologies is important as it helps to learn the techniques and resources available for understanding and applying semi-automatic Machine Learning methods for ontology construction in the future [Poveda-Villalón et al. 2010]. Additionally, having a knowledge base available is crucial for gaining a deeper understanding of the domain to which the ontology pertains [Braga et al. 2023].

Manual ontology construction can be performed using applications such as Protégé, which provides facilities for representation context, graph visualization, and modification [Musen 2015]. OWL, RDF, and RDFS are used by Protégé and humans to manually construct the ontology [Bangyal et al. 2022]. Representing an ontology in a format such as triples makes it suitable for being understood by computational processes. An interesting example is searching an ontology in this format using a procedure called SPARQL (SPARQL Protocol and RDF Query Language) [Bellini et al. 2014].

This work aims to present the ontology built manually to aggregate the knowledge of DAOs (Decentralized Autonomous Organizations), obtained manually from web pages. In addition to this Introduction, Section 2 describes the methodology used, with two subsections. Section 3 describes how to use the ontology created through Protégé using SPARQL, with some illustrative examples. In Section 4, we present the conclusion of the results and future work proposals, which include a process of creating the same ontology using Deep Learning. Finally, in Section 5, acknowledgments are made to those who supported this work.

2. Metodology

2.1. Blockchain, DAOs, DeFi and others

A Decentralized Autonomous Organization (DAO) is a form of organization based on blockchain technology that is generally governed by its members, who hold tokens [Santana and Albareda 2022]. Tokens, a type of cryptocurrency (among other meanings), can be acquired or received in some form by any person. As the owner of these tokens, the person gains the right to vote on matters directly related to the governance of the DAO. The governance rules of DAOs are characterized through computer programs known as smart contracts, which are executed and validated within the blockchain of the Ethereum network through a resource called the Ethereum Virtual Machine (EVM). The features of smart contracts, such as a distributed blockchain database, cause the rules of the organization to be enforced by the very code that defines the DAO, thus making it self-governed.

Therefore, DAOs are different from traditional organizations because they are self-governing and function autonomously in a decentralized manner without the need for intermediaries. In contrast, traditional organizations are subject to rights and responsibilities defined by the legal system of the environment in which they operate.

2.2. Introduction to the methodology

There are three important concepts to understand: **data**, **information**, and **knowledge**. Consider Figure 1 and pay attention to the fact that the graphical representation **DAO** says nothing. It is just a record of the word that represents a being, object, or phenomenon. In other words, data is something that was given, granted, or offered.

Figura 1. A datum

Such a representation tells us nothing. There is not enough characterization for us to conclude any meaning of the word **DAO**. Observing Figure 2, one can notice a characterization or amount of information that allows for a better understanding of what **DAO** means.

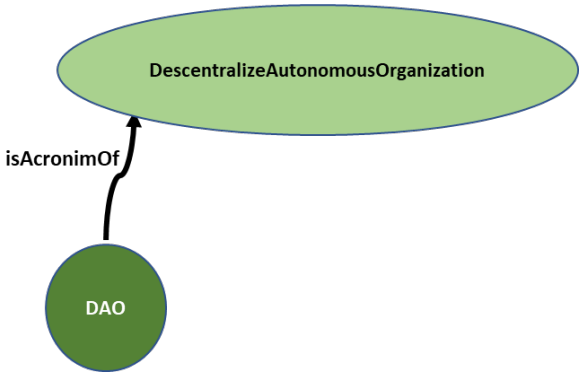

Figura 2. More data to improve a little understanding of the meaning of DAO. Em vermelho,

By adding more data to the figure above, an attempt is made to further contextualize the meaning of **DAO**. For humans, Figure 3 can improve this understanding. However, for a computer, even if it understood the graphical representation, the word **Blockchain** does not add anything to the context of **DAO**.

Knowledge is new information received by a receiver. Ontology is a data model where information is represented in a structured and organized way, aiming at knowledge generation. It is a set of related concepts. So far, we have represented ontology graphically. However, machines don't understand graphical models. Hence, there is a need to use formal ontology representation languages mentioned in the Introduction, such as RDF, RDFS, and/or OWL. Consider Figure 4[1] where **DAO** contextualization is being extended in an organized way, i.e.: "**DAO** is an acronym for Decentralize Autonomous Organization and exists (or is a class of or is a sub-class of) in the **Blockchain** environment".

Therefore, knowledge was generated, meaning that new information was received by the receiver. **Blockchain** is also closer to machine understanding, as the small dataset

[1] Available in `https://osf.io/pqmvd`

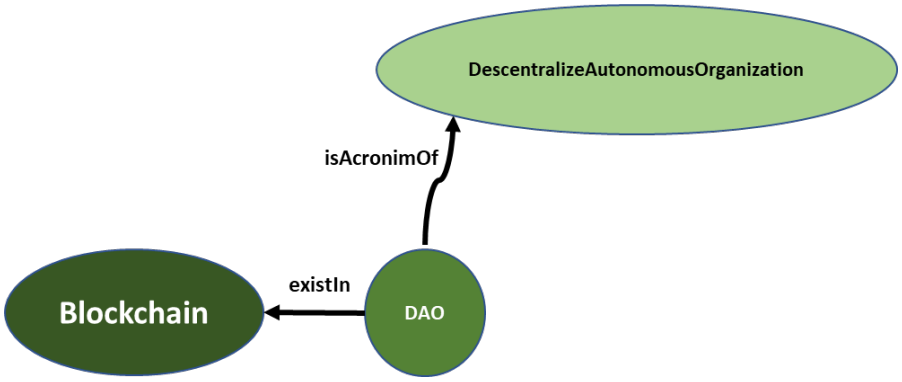

Figura 3. The notion of DAO becomes clearer, but Blockchain is not characterized.

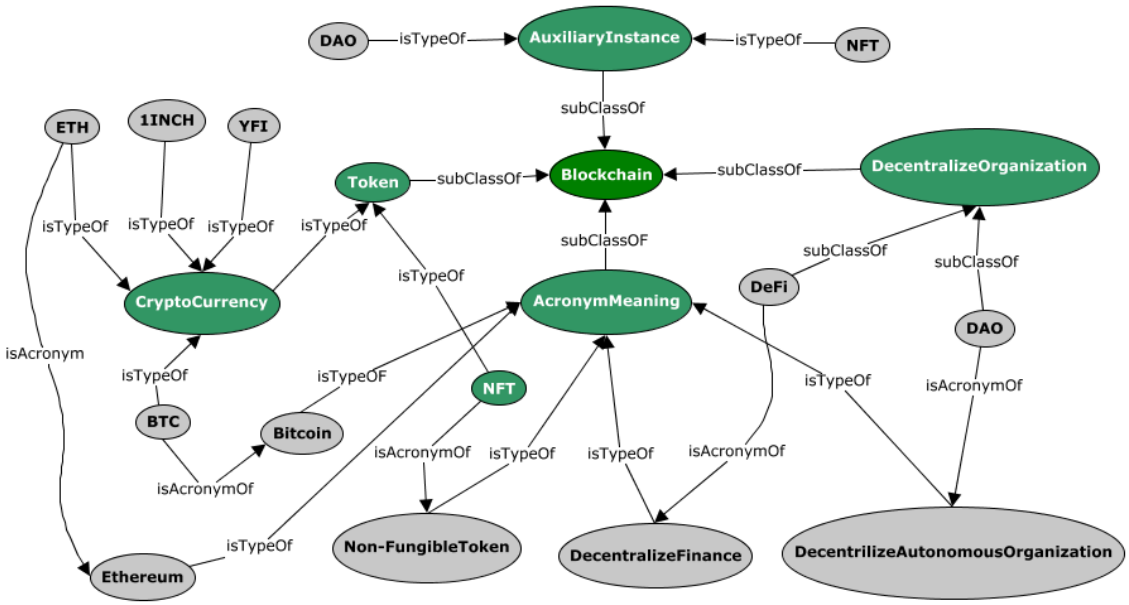

Figura 4. Partial characterization of a domain where DAO, Blockchain and others concepts are now more understandable. To build this figure, the tool CMapsTool was used [Cañas et al. 2005]

is organized and contextualized with each other, forming what has already been defined as an **ontology**. However, for machine accessibility, the ontology from above figure is represented in **triples**, as shown in Table 1.

In the context of triples in Table 1 and the graph in Figure 4, the arrow is a graphical representation of the relationship between the **subject** and the **object**, with the **predicate** being the term that distinguishes the relationship. The arrow typically points from the subject to the object, indicating the direction of the relationship (or properties) including your name.

2.3. How to build an ontology

An ontology is, also, a formal representation of knowledge within a domain, consisting of a set of concepts (classes and sub-classes), relationships (predicates, properties or relations) between those concepts, individuals (instances) that instantiate the classes and

Tabela 1. The formal representation of an ontology (partial).

Subject	Predicate	Object
Token	subclassOf	Blockchain
BTC	isTypeOf	Cryptocurrency
ETH	isTypeOf	Cryptocurrency
...	...	...
DAO	subclassOf	Blockchain
DAO	isAcronymOf	DecentralizeA...Organization
DeFi	subclassOf	DAO
Token	isManagementBy	DAO
Cryptocurrency	isTypeOf	Token
NFT	isTypeOf	Token
...	...	...

sub-classes, and axioms that define constraints and rules on the classes, properties, and individuals. The classes represent the concepts within the domain, the properties represent the relationships between those concepts, and the individuals represent specific instances of the classes. The axioms define restrictions and rules on the classes, properties, and individuals to maintain data consistency. The main axioms, all available in Protégé are:

- Hierarchical constraint
- Constraint by relationship
- constraint by cardinality
- Symmetry of relationships
- Transitivity of relationships
- Inversion
- Equivalence
- Disjunction

There are many techniques, both manual and semi-automatic, for building ontologies, and many of these techniques are described in the literature. The choice of technique often depends on the specific requirements of the ontology being built, as well as the experience and preferences of those involved in its construction [Subhashini and Akilandeswari 2011] [Isotani and Bittencourt 2015] [Farquhar et al. 1997] [Al-Arfaj and Al-Salman 2015] [Dahab et al. 2008]. The approach proposed by Thiago Castro Ferreira provides a useful set of steps for constructing an ontology, which involves the following steps: (a) determine the scope; (b) enumerate terms; (c) define classes; (d) define properties; (e) define constraints; and (f) create instances. These recommendations are fundamental when constructing the first ontologies [Ferreira 2013]. However, due to the complexities associated with the domain of the ontology, common sense and experience are often the most important resources to be used. In some cases, the tool used to build the ontology can also influence its design.

In Figure 4, the ovals in gray represent instances, while the others represent classes or sub-classes. Classes and sub-classes provide context for instances. In this case, an auxiliary class called **AuxiliaryInstance** was created specifically to meet a requirement of the Protégé tool, as it was necessary to indicate the acronyms for **DAO** and **NFT** (and potentially others). On the other hand, to indicate that **BTC** is an acronym for **Bitcoin**, this

auxiliary class was not needed, as **BTC** was already an individual. Furthermore, having a subclass such as **Organization** in the ontology allows for the inclusion of organizations other than **DeFi** and **DAO**.

2.4. How to use the Protégé to implement this graphical ontology

Referring to Figure 4, the classes and subclasses, as well as the instances and predicates, were implemented in Protégé. The result is shown in Figure 5.

Figura 5. Respectively, the hierarchy of classes and subclasses, the individuals (instances) and the relations (predicates or properties)

From the hierarchy, it can be seen that **Cryptocurrency** is a subclass of **Token**, which in turn is a subclass of the **Blockchain** class. This relationship is formally and automatically established by Protégé.

Next, we implement the instances (individuals) defined in the main graph, as shown in the second sub-figure of Figure 5. We then proceed to implement the **hasAcronym**, **isAcronymOf**, and **isManagedBy** relationships, which are visible in the third sub-figure. After that, we characterize each of the relationships and their interconnections. By doing this, we apply the predicates (relationships) to the instances. This can be seen in Figures 6 and 7.

In Figure 6, through the **Description: DAO** field, we specify that the type of the DAO instance is **AuxiliaryInstance**. On the right side, we establish that DAO is an acronym of **DecentralizedAutonomousOrganization** using the **isAcronymOf** relationship.

Figure 7 shows the **DecentralizedAutonomousOrganization** instance, which is characterized as being of the Acronym type. On the right side, Protégé's Reasoner automatically infers, highlighted in yellow, that this instance has the acronym **DAO**.

The inference was made possible because the **isAcronymOf** relationship was declared as the inverse of the **hasAcronym** relationship. This can be seen in Figure 8, where all the relationships needed to complete the ontology graph are also shown.

Thus, the complete ontology proposed was constructed and saved in the file ***decom.ttl***, which is available on the OSF platform [Braga et al. 2022a]. The topology for this example is also available in the same place, under the name ***dao.ttl***[2]. The .ttl file extension indicates that the file is in the Turtle format, which is a text-based serialization for RDF data [Beckett and Berners-Lee 2008].

[2]The ontology source is also available at the following location: `https://osf.io/gwjkh`

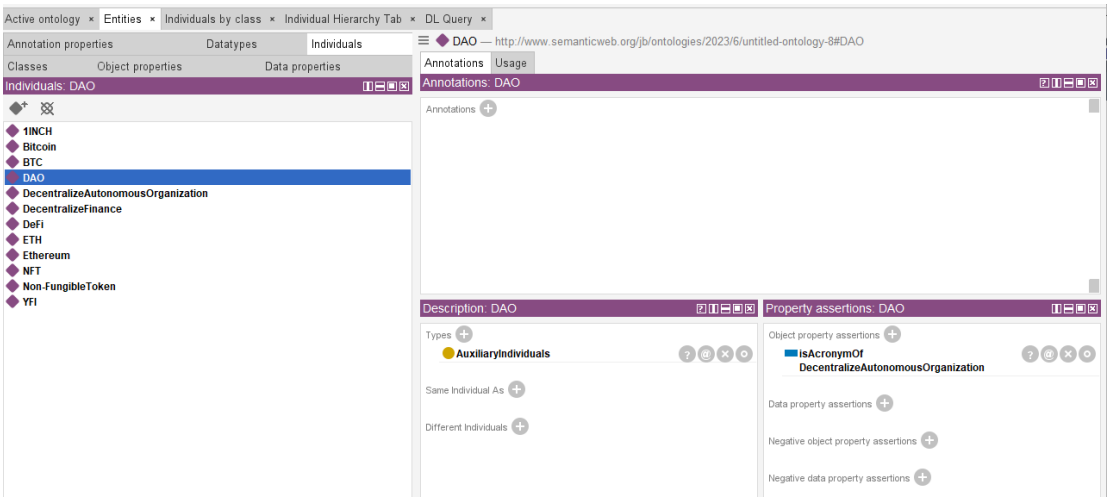

Figura 6. DAO is a type of AuxiliaryInstance and isAcronymOf DecentralizeAutonomousOrganization

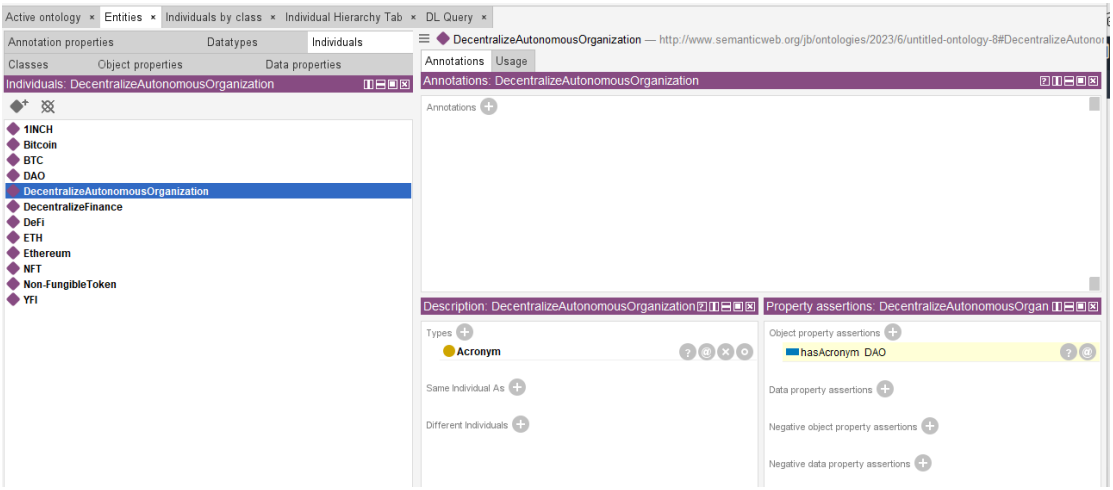

Figura 7. DescentralizeAutonomousOrganization is a type of Acronym! By inference from a Protégé reasoner, DecentralizeAutonomousOrganization hasAcronym DAO (yellowish)

3. How to retrieve knowledge featured in an Ontology

SPARQL Protocol and RDF Query Language) is a semantic query language for databases—able to retrieve and manipulate data stored in RDF format [W3C 2013a 2013, W3C 2013b 2013, W3C 2010a 2010, W3C 2008a 2008].

In simple terms, SPARQL is the query language of the Semantic Web. It is executed against RDF datasets, which consist of RDF graphs. It can: (a) retrieve values from structured and semi-structured data; (b) explore data by querying unknown relationships; (c) perform complex joins of disparate databases in a single, simple query; and (d) transform RDF data from one vocabulary to another [Feigenbaum 2009].

A SPARQL query, written in the Turtle format, is structured in the following order:

```
# prefix declarations, for abbreviating URIs
PREFIX foo: <http://example.com/resources/>
```

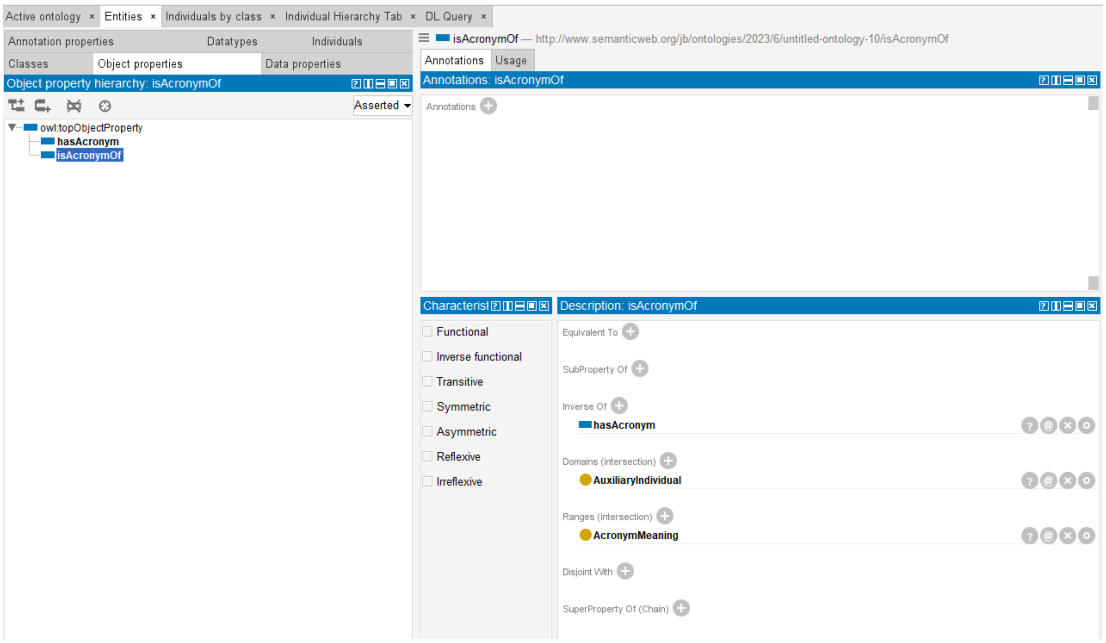

Figura 8. isAcronym is inverse of hasAcronym and can be used from AuxiliaryInstance to AcronymMeaning.

```
...
# dataset definition, stating what RDF
#  graph(s) are being queried
FROM ...
# result clause, identifying what
#  information to return from the query
SELECT ...
# query pattern, specifying what to query
#  for in the underlying dataset
WHERE {
    ...
}
# query modifiers, slicing, ordering, and
#  otherwise rearranging query results
ORDER BY ...
```

To execute a SPARQL query, you need to have access to a SPARQL endpoint. A SPARQL endpoint is a server that exposes its data via the SPARQL protocol. It is used to handle client requests and allows data to be published on the web for querying. This means that you can send SPARQL queries to the endpoint, and it will return the results of the query based on the data it has available. An SPARQL endpoint can provide access to any kind of data, not just an ontology. For example, it provides access to RDF data, which can include ontologies, but can also include other types of data such as instance data or factual information. So, while an endpoint can provide access to an ontology, it is not limited to doing so.

You can also create your own SPARQL endpoint using Apache Jena[3]. Jena includes various command-line utilities that can help you with a variety of tasks in developing Jena-based applications.

Below are three examples of SPARQL queries that can be run on the dao.ttl file, which contains the ontology created for this text. These queries demonstrate the variety of results that can be obtained from the ontology, among many other possibilities.

Query 1 List all Acronym Meaning:

The query:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/>

SELECT ?individual
WHERE
?individual a :AcronymMeaning .
```

1
<http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/DecentralizeFinance>
2
<http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/Non-FungibleToken>
3 <http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/Ethereum>
4 <http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/Bitcoin>
5 <http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/DecentralizeAutonomousOrganization>

Explanation: This query uses the PREFIX declaration to define the base URI of the ontology[4], and the **SELECT** statement to specify the variable **?individual** that will be returned in the query results. The **WHERE** clause contains a triple pattern that matches all individuals of the class **AcronymMeaning**. The a keyword is shorthand for the **rdf:type** property, which is used to specify the type (i.e., class) of an individual.

Query 2 Same listing, minus the URIs in the result:
The query (without the header):

```
SELECT (REPLACE(STR(?individual), ".*(#—/)", ) AS ?name)
WHERE
?individual a :AcronymMeaning .
```

Result of the query:

1 DecentralizeFinance
2 Non-FungibleToken
3 Ethereum

[3]https://jena.apache.org/tutorials/sparql.html
[4]PREFIX : http://www.semanticweb.org/jb/ontologies/2023/6/untitled-ontology-10/

4 Bitcoin
5 DecentralizeAutonomousOrganization

Explanation: In this query, the **STR** function is used to convert the **URI** of the individual into a string, and the **REPLACE** function is used to remove everything before the last **#** or **/** character in the **URI**, leaving only the local name of the individual. The result is returned in a variable named **?name**, which is specified using the AS keyword.

When you run this query on a SPARQL endpoint that has loaded the ontology, it will return a list of local names of all individuals that are instances of the class **AcronymMeaning**.

Query 3 Here is an example of a SPARQL query that uses the COUNT function to retrieve information from the ontology:

The query (without the header):
SELECT (COUNT(?individual) AS ?count)
WHERE
?individual a :AcronymMeaning .

Result of the query:
1 "5"|http://www.w3.org/2001/XMLSchema#integer>

Explanation: This query uses the **COUNT** function to count the number of matching individuals, and returns the result in a variable named **?count**. The rest of the query is similar to the previous examples, with a **WHERE** clause that contains a triple pattern matching all individuals of the class **AcronymMeaning**.

Query 4 Using the ontology stored in decom.ttl, run a query to count the number of DAOs that are characterized.

The query (note the header):
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX decom: <https://ghaia.pt/kb/decom.ttl#>

SELECT (COUNT(?dao) AS ?count)
WHERE
?dao rdf:type decom:DAO .

Result of the query:
1 "202"|http://www.w3.org/2001/XMLSchema#integer>

Explanation: This query uses the **COUNT** function to count the number of individuals that are of type **decom:DAO**. The **?dao** variable represents each individual **DAO**, and the **?count** variable represents the total count of **DAO** individuals.

4. Conclusions and future work

Ontology is a relatively new field of scientific research, and much work is still needed to make it accessible to domain experts in specific areas of knowledge. If applied, ontology can bring significant benefits to these domains.

Ontologies can be used to represent knowledge in a wide range of domains, and many different fields can benefit from their use. Some examples of domains that can benefit from ontology include:

- Medicine and healthcare: Ontologies can be used to represent medical knowledge, such as diseases, symptoms, treatments, and drugs, and can help with tasks such as diagnosis, treatment planning, and drug discovery.
- Biology: Ontologies can be used to represent biological knowledge, such as genes, proteins, pathways, and interactions, and can help with tasks such as data integration, analysis, and interpretation.
- E-commerce: Ontologies can be used to represent product knowledge, such as product categories, features, and relationships, and can help with tasks such as product search, recommendation, and comparison.
- Natural language processing: Ontologies can be used to represent linguistic knowledge, such as word meanings, relationships, and usage patterns, and can help with tasks such as text understanding, generation, and translation.

Much work still needs to be done, and the following are some topics that deserve attention:

i. The ontology created using Protégé and stored in ***decom.ttl*** must be evaluated, checked, and compared with other similar ontologies to ensure its validity. However, the manual process of building an ontology can be time-consuming and exhausting, and may not always produce accurate results. As such, it is recommended to use semi-automatic techniques that involve a combination of manual input and automated processes to develop and update the ontology. These techniques, which are constantly being improved, generally involve the use of deep learning and text capture from the web [Al-Aswadi et al. 2020].
ii. Ontology alignment is the process of finding correspondences between concepts in different ontologies, and it is an important research problem with applications in various fields such as data integration, data transfer, and data preparation. Many state-of-the-art ontology alignment systems use domain-dependent approaches with handcrafted rules or domain-specific architectures, which can make them unscalable and inefficient [Iyer et al. 2021].
iii. Artificial intelligence techniques, such as machine learning and natural language processing, can be used to automatically extract ontological knowledge from various sources, such as text corpora, databases, and the web. This process is known as ontology learning, and it is an active area of research in the field of ontology.
iv. Ontology evaluation: This involves assessing the quality of ontologies with respect to various criteria, such as their consistency, completeness, and usability.
v. Ontology-driven information retrieval: This involves using ontologies to improve the effectiveness of information retrieval systems by providing a richer representation of the knowledge within a domain.

Authors will be interested in progressing in areas that meet items i and iii.

5. Acknowledgments

Many of the concepts and ideas presented in the Methodology section were adapted from the lecture of [Ferreira 2013], to whom the authors are grateful. This work fulfills the second and final requirement proposed during the studies conducted at the Oscar Sala Chair, under the coordination of Prof. Dr. Virgílio Almeida and the group led by Prof. Dr. Paola Guerra [Braga et al. 2022b]. The authors would like to thank them for the academic opportunity.

Referências

Al-Arfaj, A. and Al-Salman, A. (2015). Ontology construction from text: challenges and trends. *International Journal of Artificial Intelligence and Expert Systems*, 6(2):15–26.

Al-Aswadi, F. N., Chan, H. Y., and Gan, K. H. (2020). Automatic ontology construction from text: a review from shallow to deep learning trend. *Artificial Intelligence Review*, 53:3901–3928.

Bangyal, W. H., Rehman, N. U., Nawaz, A., Nisar, K., Ibrahim, A. A. A., Shakir, R., and Rawat, D. B. (2022). Constructing domain ontology for alzheimer disease using deep learning based approach. *Electronics*, 11(12):1890.

Beckett, D. and Berners-Lee, T. (2008). Turtle - Terse RDF Triple Language. `https://www.w3.org/TeamSubmission/turtle/`. Accessed: 2023-07-06.

Bellini, P., Nesi, P., and Venturi, A. (2014). Linked open graph: browsing multiple sparql entry points to build your own lod views. *Journal of Visual Languages & Computing*, 25(6):703–716.

Braga, J., Regateiro, F., and Stiubiener, I. (2022a). Human-algorithm: Governance. Acessed in 07/07/2023.

Braga, J., Regateiro, F., Stiubiener, I., and Braga, J. C. (2022b). A proposal to improve research in ai algorithm and data governance. Portuguese version: `https://osf.io/xcpsd`.

Braga, J., Regateiro, F., Stiubiener, I., and Braga, J. C. (2023). Governance of a dao for facilitating dialogue on human-algorithm interaction and the impact of emerging technologies on society. In Cantarini, P., editor, *IA e Espistemologias do Sul*, pages 1–28. Editora Lumen Juris Direito, Sao Paulo, SP Brazil. To be publisched. Available in preprint by OSF Preprints, DOI: 10.31219/osf.io/wexjr.

Cañas, A. J., Carff, R., Hill, G., Carvalho, M., Arguedas, M., Eskridge, T. C., Lott, J., and Carvajal, R. (2005). Concept maps: Integrating knowledge and information visualization. *Knowledge and information visualization: Searching for synergies*, pages 205–219.

Dahab, M. Y., Hassan, H. A., and Rafea, A. (2008). Textontoex: Automatic ontology construction from natural english text. *Expert Systems with Applications*, 34(2):1474–1480.

Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: A tool for collaborative ontology construction. *International journal of human-computer studies*, 46(6):707–727.

Feigenbaum, L. (2009). SPARQL By Example: A Tutorial. https://www.w3.org/2009/Talks/0615-qbe/. Accessed: 2023-07-06.

Ferreira, T. C. (2013). Introdução a ontologias e à web semantica. Availeble: \url{https://www.youtube.com/playlist?list=PLLrlHSmC0Mw6lEj060psXrt3BSATBFVzV}.Acessed:05/07/2023.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220.

Isotani, S. and Bittencourt, I. I. (2015). *Dados abertos conectados*. Novatec Editora, São Paulo, SP, Brasil.

Iyer, V., Agarwal, A., and Kumar, H. (2021). Veealign: multifaceted context representation using dual attention for ontology alignment. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10780–10792.

Kietz, J.-U., Maedche, A., and Volz, R. (2000). A method for semi-automatic ontology acquisition from a corporate intranet. In *EKAW-2000 Workshop "Ontologies and Text", Juan-Les-Pins, France, October 2000*.

Mishra, S. and Sarika, J. (2014). Automatic Ontology Acquisition and Learning. *International Journal of Research in Engineering and Technology*, 03(14):38–43.

Musen, M. A. (2015). The protégé project: a look back and a look forward. *AI matters*, 1(4):4–12.

Poveda-Villalón, M., Suárez-Figueroa, M., and Gómez-Pérez, A. (2010). A Double Classification of Common Pitfalls in Ontologies. *Proceedings Workshop on Ontology Quality - OntoQual 2010. CEUR Workshop Proceedings*, pages 1–12.

Santana, C. and Albareda, L. (2022). Blockchain and the emergence of decentralized autonomous organizations (daos): An integrative model and research agenda. *Technological Forecasting and Social Change*, 182:1–15.

Subhashini, R. and Akilandeswari, J. (2011). A survey on ontology construction methodologies. *International Journal of Enterprise Computing and Business Systems*, 1(1):60–72.

W3C 2008a (2008). SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/. Accessed: 2023-07-06.

W3C 2010a (2010). SPARQL New Features and Rationale. https://www.w3.org/TR/sparql-features/. Accessed: 2023-07-06.

W3C 2013a (2013). SPARQL 1.1 Overview. https://www.w3.org/TR/sparql11-overview/. Accessed: 2023-07-06.

W3C 2013b (2013). SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/. Accessed: 2023-07-06.