**Wooftology: A Knowledge Graph / Ontology defining Dog Breeds**



Master of Science (MS) in Data Science  
Module: ITC6005A1 – Knowledge Graphs  
Instructor: Dr. Drakakis Georgios

Students’ names & IDs:

Alkiviadis Kariotis 241735

Maria Varna 277825

Ioanni Tsaka

Term: Spring Term 2024  
Type: Group Term Project - Report

Submission Date: Sunday 28th of July

Word Count: ±4200

Table of Contents

[Abstract 3](#_Toc173056978)

[Keywords 3](#_Toc173056979)

[Introduction 4](#_Toc173056980)

[MATERIAL AND METHODS 6](#_Toc173056981)

[Data Sources and Software 6](#_Toc173056982)

[Preprocessing and handling Dog data 7](#_Toc173056983)

[Ontology Classes – Building the Ontology 11](#_Toc173056984)

[Object Properties 15](#_Toc173056985)

[Data Properties 16](#_Toc173056986)

[Individuals 17](#_Toc173056987)

[Results & Discussion 18](#_Toc173056988)

[Value Partitions 18](#_Toc173056989)

[Defined Classes & Restrictions 19](#_Toc173056990)

[Queries 19](#_Toc173056991)

[Conclusion and Future Work 24](#_Toc173056992)

[References 25](#_Toc173056993)

[Appendices 26](#_Toc173056994)

[Appendix 1 26](#_Toc173056995)

[Appendix 2 29](#_Toc173056996)

[Appendix 3 30](#_Toc173056997)

[Appendix 4 31](#_Toc173056998)

[Appendix 5 33](#_Toc173056999)

[Appendix 6 37](#_Toc173057000)

[Appendix 7 38](#_Toc173057001)

# Abstract

The project aims at creating an ontology based on dog breeds. The goal is to define and at the same time categorize the various dog breeds, based on each breed's characteristics. Specifically, the ontology encompasses a comprehensive knowledge graph structure containing information for every single breed, which was originally sourced through a Kaggle dataset and then reconstructed to fit the needs of the specific project. The ontology is centered around dog breeds and their characteristics, defining them as main classes, and then setting their characteristics such as size, intelligence etc. as subclasses. SPARQL was used to create sufficient queries, answer natural language questions and extract specific information for breed’s qualities and needs. Individuals were inserted to enrich the ontology by providing examples of dogs. Generated graphs through Protege, WebVowl and GraphDB were used, to enhance the visual representation of the ontology and provide a better understanding of the whole structure.

## Keywords

Knowledge Graph, Ontology, Dog Breeds, Protege, SPARQL Queries, Classes, Characteristics, Properties

# Introduction

The internet contains vast amounts of semantically isolated data that users can access and understand independently. The creator of the Word-Wide Web, Tim Berners-Lee introduced the Semantic Web, an evolutionary idea that all the available information could not only be connected, but also understood by the Web (Berners-Lee,T., et al., 2001). Berners-Lee in his work emphasized the development of technical concepts like Uniform Resource Identifiers (URI), Resource Description Framework (RDF) and Web Ontology Language (OWL). Recently, the concept of knowledge graphs (KG) has been widely explored by researchers to access information systems containing structured knowledge (Kuck G., 2004). This graph-based representation of knowledge contains nodes that represent entities, linked to each other by edges that represent relationships. The relationships can be organized into schemas or ontologies to better interpret the represented data (Zou X., 2020). Evolving as models for describing and querying heterogeneous data, knowledge graphs appear to have real-world semantics. Semantics rely on the domain, a concept emerged by ontologies. The domain of interest is best described by the structure of the ontology, which contains entities(nodes), properties or predicates (edges), constraints and axioms all contributing to a comprehensive, directed graph (Kejriwal M., 2022). Into the concept of a knowledge graph, every single node, ranging from the subject to the predicate and to the object, is defined and accessed by a Uniform Resource Identifier (URI). This is a unique string of characters, aiming to identify a particular source on the internet (Berners-Lee, Fielding, & Masinter, 2005).

Data exchange and semantic integration lies on the concept of Open World Assumption (OWA). OWA implies that the absence of a fact or a data does not indicate that the fact is wrong, it simply indicates that the current source or the schema mapping include arbitrary facts (Libkin L. and Sirangelo C., 2009). To avoid overlapping in certain conceptual areas, foundational ontologies were introduced to cover a detailed framework for specific types of entities. These high-level and domain independent ontologies were created to establish a universal base to prevent ambiguities and evolve matching systems. For example, DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) aims to capture the semantics in natural language and human cognition by distinguishing the difference between endurant and perdurant entities (Schmidt et al., 2018).

Ontologies can serve as knowledge bases for a great variety of topics. This project focuses on an innovative theme based on man’s best friend. As society is evolving, dogs and all animals in general gain more respect as creatures and serve not only as a great companion but in many cases as service dogs. By categorizing breeds and their characteristics, it is possible to gain more insights into their behavior, habits and care requirements. The created ontology can provide potential owners with information about the breeds and help them develop a right and responsible preference on a dog. Thus, it promotes reasonable ownership and offers great intel on the rich variety of these beautiful beings.

As in any other instance nowadays, when it comes to similar work, you can find anything on the internet. The most famous and widely used ontology, in knowledge graph format, out there is the DBpedia Ontology. DBpedia is a general-purpose knowledge base, which contains ontologies and knowledge graphs for numerous and different domains. In general, DBpedia extracts structured information from the Wikipedia website, and organizes that information into knowledge graphs format. When it comes to dog breeds, their ontology provides detailed descriptions of different dog breeds, their physical attributes, and behaviors. Also, it even contains information regarding each dog’s dietary adaptations, their evolutionary background and even for cultural significance (Dog, n.d.).

For all the above purposes, the project offers a great variety of information regarding dog breeds. The breeds and their own characteristics were defined as the main classes and the ontology was built around them. Afterwards, object and data properties were defined to ensure the relationships between instances and individuals. Restrictions and constraints assured that the relationships presented logical connections and enabled precise definitions for the ontology. To further enrich the ontology, individuals were created and imported and then used as examples, to make sure that the structure was solid. At the final stage, SPARQL Queries was used, to explore the ontology on a deeper level. Queries were able to answer specific natural language questions, either to ontology-class level or even for the imported individuals. So, the ability to use a combination of different restrictions, and even being able to capture any node connection in the ontology by the respective URI, provided the ability to answer “any” question regarding the dog breeds contained in the ontology.

# MATERIAL AND METHODS

## Data Sources and Software

The ontology was built based on a dataset derived from Kaggle, a widely known data source (*Dog Breeds Ranked 🐕🐾*, 2024). The data provided essential information about dog breeds and their characteristics. While analyzing the data, a basic hierarchy of the characteristics was established, encompassing a variety of their physical traits, cost, genetic lineage and needs.

The ontology was developed using the Protege software. Protege is an open-source, ontology development software which helped in engineering classes, properties and the relationships between them. Its tools provided a comprehensive framework for extracting insights from the set. Also, for the Queries we used the internal tool of Protege, called SPARQL Query tab, which provided a space to formulate the desired query but also a space for the visualizations of the results which the query retrieved. Finally, for visualizations purposes Graph DB, WebVowl and OntoGraf were used to develop some visuals to better demonstrate and at the same time understand the knowledge structure. In Figure 1 is presented the completed produced ontology.

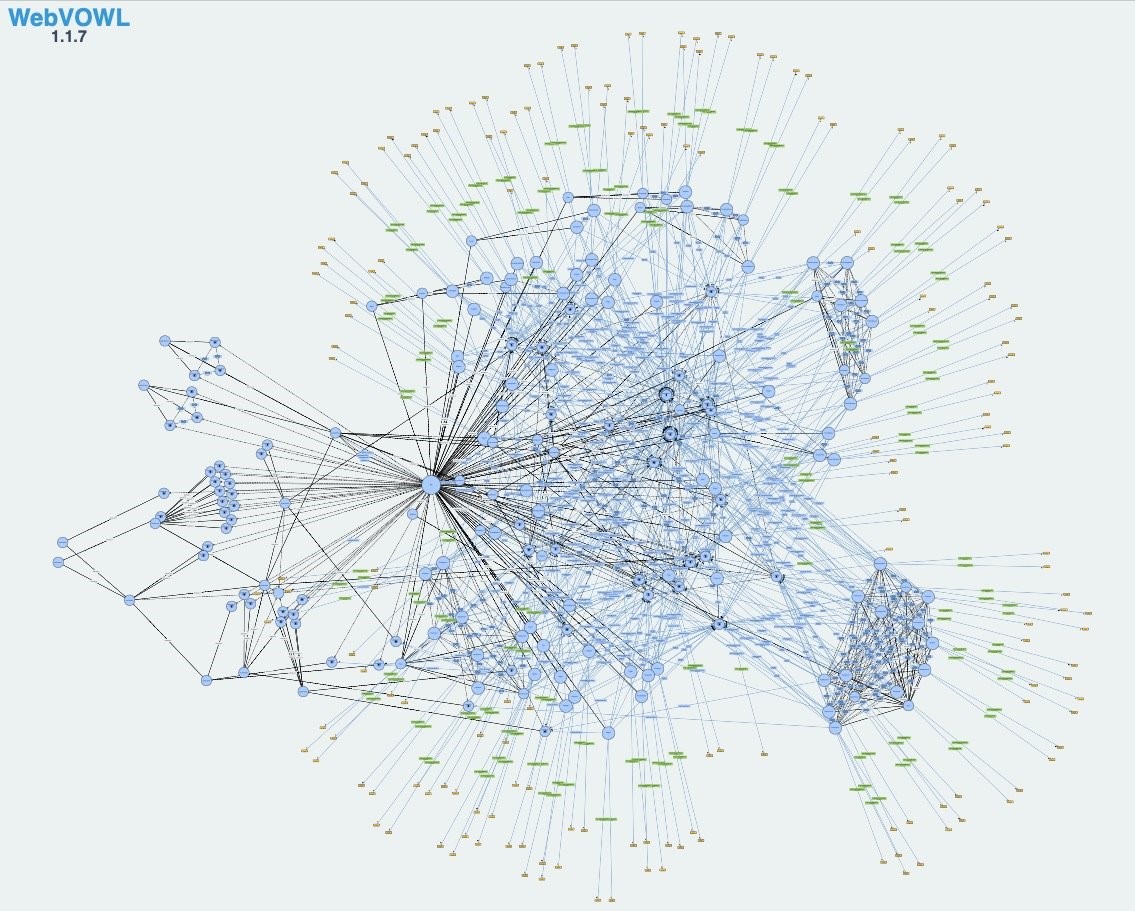


Figure 1: Final ontology of Dog breeds, showcasing all the nodes from classes to individuals as well as their predicates.

## Preprocessing and handling Dog data

As stated above, the Kaggle dataset contributed to the early stages of the ontology building. Specifically, some of the columns were excluded according to the desired size of the ontology. Of the total nineteen columns originally presented in the csv dataset, eleven were used for the specific ontology. The used columns (that were turned into classes) together with their explanations and logic in more depth are:

1. Breed: Containing the name of each dog breed. In total there were eighty-seven distinct breeds. Here there were breeds that had some further categorization such as:

* Terrier
  + Australian
  + Bedlington
  + Border
  + Boston
  + Bull
  + Cairn
  + Dandie Dinmont
  + Kerry Blue
  + Norfolk
  + Scottish
  + Staffordshire Bull
  + Tibetan
  + West Highland
  + Yorkshire

Also, for the aforementioned subclass of dog there is a graph in Figure 2.

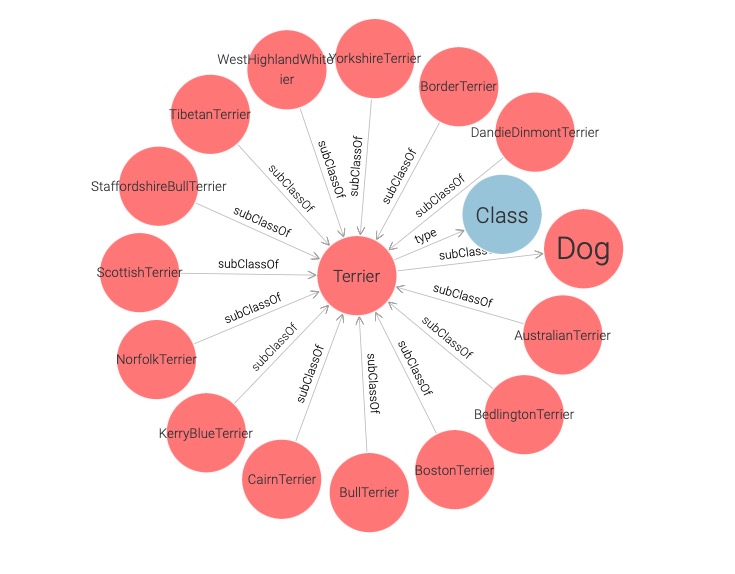


Figure 2: GraphDB visual of the subclass Terrier and its subclasses

So, the one’s which were able to be grouped together, they were assigned under a third depth class under the Dog Class (Dog 🡪 Terrier 🡪 categories). For all the categorizations done, see Appendix 1.

1. type: The category or group that each breed belongs to such as:

* Terrier
* Working
* Herding
* Hound
* Toy
* Sporting
* Non-sporting

1. popularity ranking: This is an integer indicating the popularity of each breed, and it was used in the Data properties section for the individuals.
2. size: The size category of each breed. Here three categories were created:

* Small Size
* Medium Size
* Large Size

1. intelligence: The intelligence level of each breed, categorized in scaled order as:

* LowerstIntelligence
* FairIntelligence
* AverageIntelligence
* AboveAverageIntelligence
* ExcellentIntelligence
* BrightestIntelligence.

1. Genetic ailments: Common genetic health issues associated with each breed. That column contained genetic ailments separated by comma and written in string format. They were manually separated and grouped into categories based on the body sections the problem or deficiency appeared:
   * GeneticAilmentsCharacteristic
     + BloodProblems
       - Anemia Blood Problems
       - Blood Vessel Problems
       - Clotting Blood Problems
       - Hemophilia Blood Problems

* BoneProblems
* Elbows Bone Problems
* Hip Bone Problems
* Knee Bone Problems
* Spinal Bone Problems
* FaceProblems
* Ears Problems
  + Deafness Ears Problems
* Eyes Problems
  + Cataracts Eyes Problems
  + Dryness Eyes Problems
  + Other Eyes Problems
* Lion Jaw Face Problems
* Nose Problems
  + Sinus Nose Problems
* Trachea Face Problems
* OrgansProblems
* Breathing Organs Problems
* Fatal Stomach Bloat Organs Problems
* Heart Defects Organs Problems
* Kidneys Organs Problems
* Liver Organs Problems
* Pancreas Organs Problems
* OtherProblems
* Birth Defects Other Problems
* Complex Immune Other Problems
* Connective Tissue Other Problems
* Dwarfism Other Problems
* Enzyme Deficiency Other Problems
* Epilepsy Other Problems
* Hair Loss Other Problems
* Meningitis Other Problems
* Nerves Other Problems
* Skin Other Problems
* Urinary Stones Other Problems
* Zinc Metabolism Other Problems
* None

1. Lifetime Cost: The estimated lifetime cost of each breed, expressed in dollars. This was used as an integer indicating the cost of each breed (in the Data properties section) and also as a class with the following categories:

* Cheap Lifetime Cost
* MediumLifetime Cost
* Expensive Lifetime Cost

1. Intelligence Rank: The intelligence rank of each breed. This is an integer indicating the intelligence ranking of each breed, and it was used in the Data properties section for the individuals.
2. Longevity(years): The average lifespan of each breed, expressed in years. This was used as an integer indicating the lifespan of each breed (in the Data properties section) and also as a class with the following categories:

* Small Longevity
* Medium Longevity
* Long Longevity

1. Grooming Frequency: How often each breed requires grooming, where three categories were used

* Daily Grooming
* Once Per Week Grooming
* Once In Few Weeks Grooming

1. Suitability for children: The suitability of each breed for families with children. The original dataset had a scaled metric (1 = high suitability, 2= medium suitability, 3= low suitability), which was respectively turned into three categories:

* Suitable for Children
* Moderately Suitable for Children
* Non-Suitable for Children

## Ontology Classes – Building the Ontology

As a first step, the two main classes created in protege were Dog and Characteristics. These two classes include the most attributes and constitute an accurate starting point for building the hierarchy. After setting the general classes, subclasses were created to expand the depth of the ontology. All the categorization from the columns that was aforementioned aided towards the creation of the subclasses of the Ontology.

In the Figure 3, there is a visual of both of the two main classes and all the smaller ones that fall “under” each one of them.

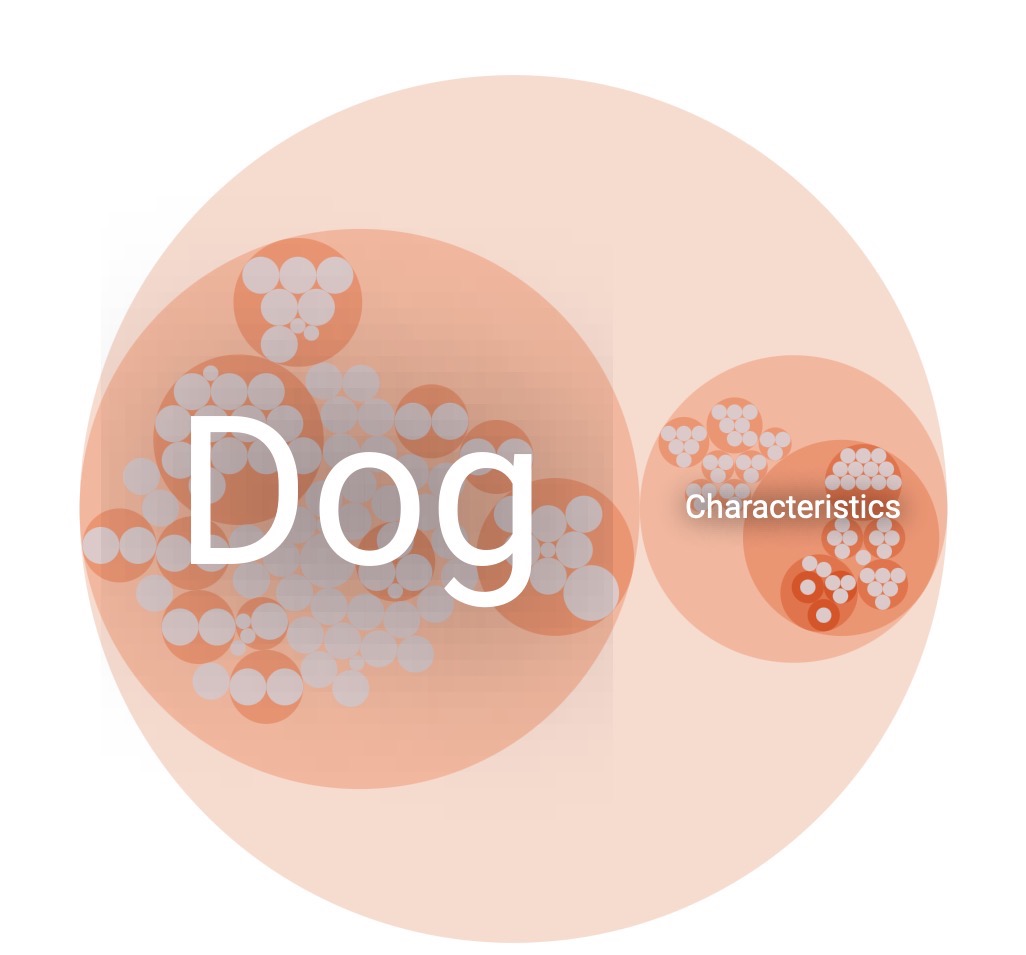


Figure 3: GraphDB visual, showing the two main classes, Dog and Characteristics. Inside them are all the other primitive and defined classes.

In order to gain a better understanding of the structure of each one, more in-depth visuals were created. For the Dog class, a sample of the classes’ hierarchy was used to visually describe it, and it can be found in Figure 4.

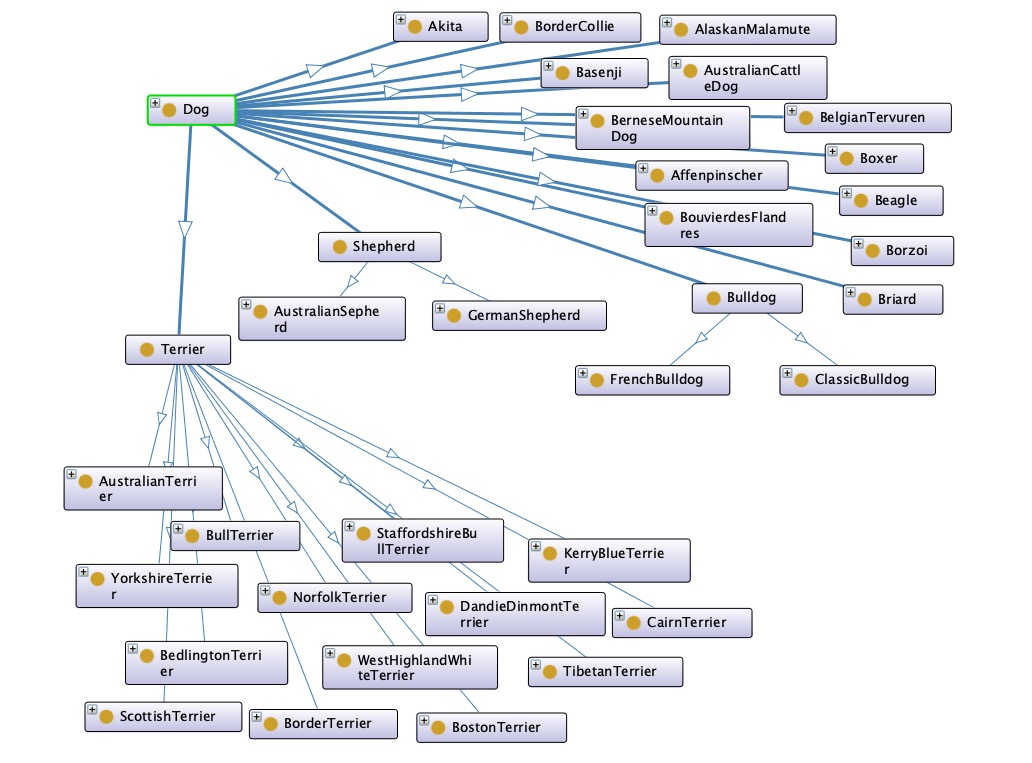


Figure 4: OntoGraf visual, representing a sample of subclasses of Dog, together with their subclasses.

In Figure 5, goes even deeper in the hierarchy, showing the connections for a specific Dog Breed, the Akita one. In the specific figure, an individual is depicted, which be explained later on.

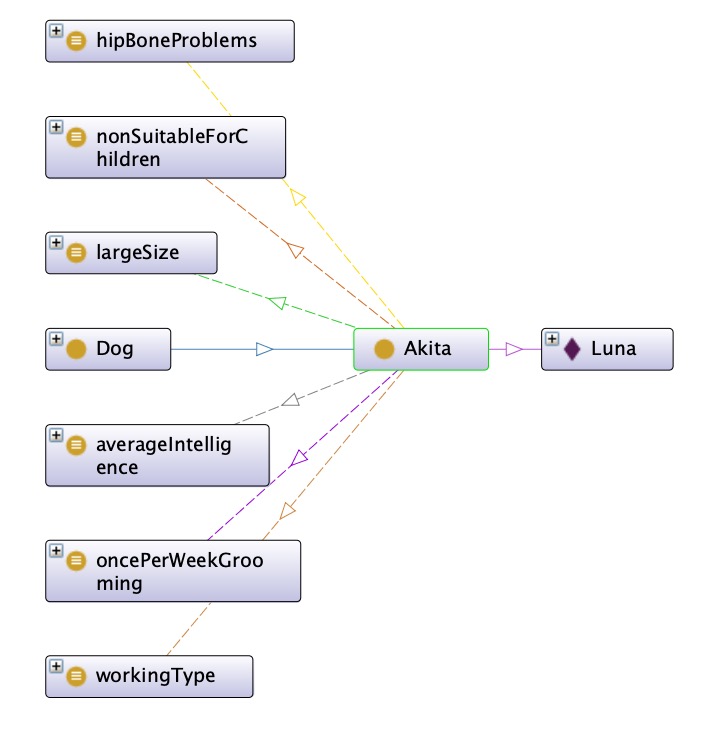


Figure 5: OntoGraf visual, showing a single Dog Breed Class (Akita) and all the connections made with Characteristics, together with an Individual.

Last but not least, in the Figure 6, there is the class hierarchy of the Characteristics class. Another visual for this can be found in the Appendix 2 (Figure 12) and at the same time a table (Table 1) containing all the subclasses.

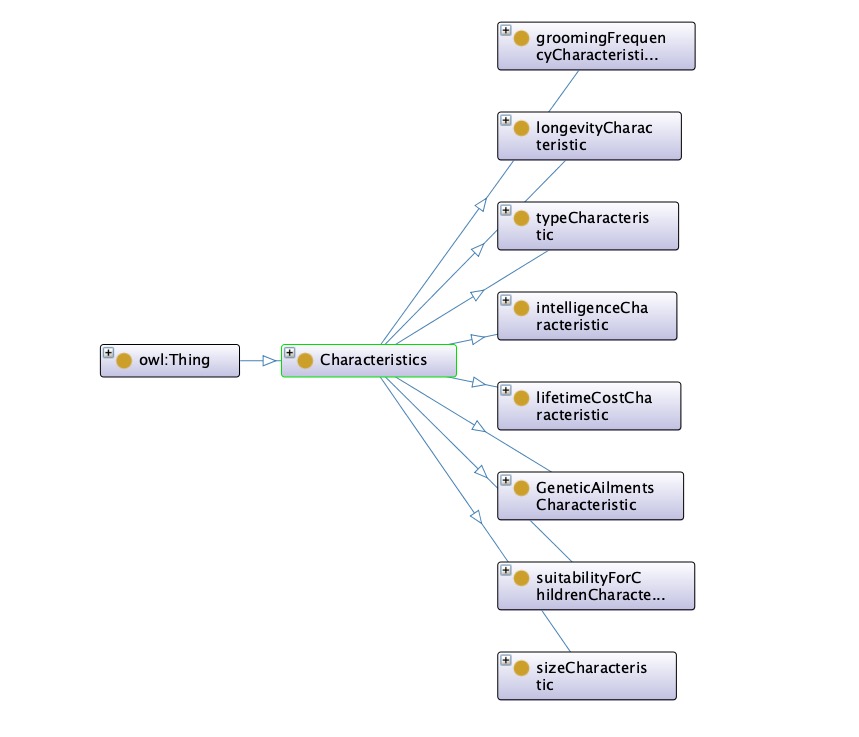


Figure 6: OntoGraf visual, showing all the subclasses of the Characteristics main class.

## Object Properties

The next step in the ontology was to define the object properties. In knowledge representation, object properties have a key role, since they contribute to defining the relationships between classes, subclasses and iteratively to individuals. The object properties used provided a well-structed ontology by revealing and specifying the interactions between classes and instances. The goal for each object property was to associate each breed with their unique characteristics and provide as much information for each breed as possible.

So, the Object Properties created are:

* areDefining: was entered to explicitly state that Characteristics are defining a Dog. The domain was set to Characteristics and the range to Dog. By the use of that property, the Open World Assumption was supported, by protecting that all the Characteristics are defining the specific entity called Dog.
* hasCharacteristics: was entered into ObjectProperties, with Domain Dog and Range the class of Characteristics. A set inversing these two ObjectProperties was manufactured, indicating that they hold an opposite semantic relationship, aiding again towards the closure of the world.
* hasGeneticAilment: an object property specified that Dog, as a Domain, can range into all the Genetic Ailment sublclasses(e.g. none, faceProblems, boneProblems,bloodProblems).
* hasGroomingFrequency: refers to a Dog as a domain must have the groomingFrequencyCharacteristic (range).
* hasIntelligence: specifies that a dog has intelligenceCharacteristic (range).
* hasSize: states that a dog must be specified by a defined size (range).
* hasSuitabilityForChildren: as a property describes whether or not a dog is suitable for children and has dog as a domain and suitabilityForChildren as a range.
* hasType: also has dog as a domain and hasTypeCharacteristic as a range and states that each breed must belong to a subclass of the typeCharacteristic class (range).

The object properties for Grooming, Intelligence, Size, Suitability for Children and Type were set as functional, because they served as Value Partitions, meaning that each entity that they will be applied on, will be able to gain/be associated with one value or one instance only, from the respected category.

## Data Properties

Similarly, the Data properties hierarchy was determined on the ontology. Datatype properties connect individuals to data values, such as strings,numbers etc. These properties define attributes of individuals by assigning literal values from predefined data types.

Every breed was assigned to several genetic ailments. For example, Border Collie presented two: Ear and Eye problems. So, a data property hasGeneticAilment was created to match each breed to the corresponding number. The data value assigned was an integer(xsd:integer). To better define the Intelligence Rank for each breed the data property hasIntelligenceRank was inserted into the ontology. The scale to measure the intelligence rate contained five subclasses: above average intelligence, average intelligence, brightest intelligence, excellent intelligence, fair intelligence and lowest intelligence. Additionally, the cost for each breed was presented by the hasLifetimeCost data property. A scale to adapt the cost into sub properties to better represent them was as following:

So, the Data properties created are as followed:

* hasGeneticAilments : is used to define the number of Genetic Ailments an individual dog has. This is in a numerical format; hence the range was set to xsd:integar.
* hasIntelligenceRank : this datatype property was used for the population of individuals to help determine the Intelligence of a Dog individual. It is in a xsd:integar , since it can only have numerical values.
* hasLifetimeCost : this datatype property was used to link the dog individuals with an estimated Cost throughout their lifespan. There are three sub properties spreading from this property:
* hasLifetimeCostCheap
* hasLifetimeCostExpensive
* hasLifetimeCostMedium
* hasLongevity : this datatype property was created to better define the time of each individual dog’s lifespan. The range was det at xsd: integer since it matched numerical values. Three sub properties were also created to develop a ranking regarding longevity:
* hasLongevityLong
* HasLongevityMedium
* hasLongevityShort
* hasPopularityRanking : this datatype property ranks the popularity of each individual dog and ranges as xsd:integer

In the original set the costs appeared in a wide range, therefore, to better categorize them a scaling was initiated for both the hasLifetimeCost and for hasLongevity, which are further analyzed in the Appendix 3 (Table 2 & Table 3).

## Individuals

Individuals serve as specific entities or instances that helped populate the ontology. Individuals belong to one or more classes and contributed towards better understanding the structure of the ontology and how the characteristics are linked to entities. In order to produce the individuals, we used Chat GPT, where we prompted it to generate an excel file, with specific structure in terms of columns, and specific range values within each column. So, we had an excel file with the following columns:

* Dog: Random name of Dog Individual, type string.
* Types: Random Dog breed (from the dog breeds that we used in our ontology), type string.
* hasGeneticAilments: an integer number tailored to the specific breed information (based on our original dataset) that every individual was assigned to.
* hasIntelligenceRank: an integer number tailored to the information for specific breeds through our original dataset.
* hasLifetimeCost: an integer number tailored to the information for specific breeds through our original dataset.
* hasLongevity: an integer number tailored to the information for specific breeds through our original dataset.
* hasPopularityRanking: an integer number tailored to the information for specific breeds through our original dataset.
* Comment: string type text with a comment with the structure: “Individual\_Name is an individual Dog of a Type Individual\_Breed”.

In total 78 individuals were produced and imported in the ontology using the rule that can be found in Appendix 4 (Figure 13), together with a visual representation (Figure 14). Through this we were able to infer information and create new nodes/knowledge.

# Results & Discussion

## Value Partitions

To further enhance the clarity and consistency across data, value partitions were explored to better model and specify the description of the Chracteristics classes. In the class of Intelligence Characteristic all the subclasses are disjoint with each other. For example, a breed cannot have averageIntelligence and at the same time lowestIntelligence. Similarly, the subclasses of lifetimeCostCharacteristic are semantically opposite and it’s impossible for a breed to haslifetimeCost as property that ranges in more than one of the subclasses. In the same way, longevityCharacteristic represents a specific Characteristic and breeds can be categorized either in long, medium or short Longevity. Additionally, breeds can only be categorized in one size subclass, creating a value partition in disjoint large, medium and small classes. Moreover, the subclasses of suitabilityForChildrenCharacteristic class are disjoint with each other and the object property used to categorize them strictly in one of the subclasses. Every breed belongs only to one type of breed. For example, a dog is impossible to be described as herdingType and ToyType simultaneously, creating one more Value Partition.

One visual example for Value Partition, and more specifically about size, can be seen in Figure 7.

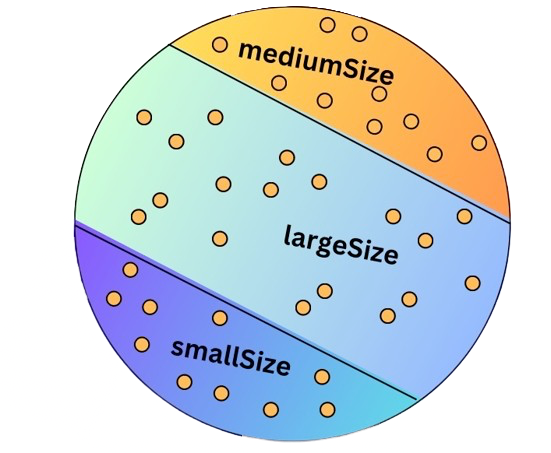


Figure 7: Handmade visual for Value Partition of sizeCharacteristic class.

## Defined Classes & Restrictions

To “fit” the object and data properties in the appropriate classes, restrictions and constraints were used. Cardinality restrictions were created to define that a property can relate to an entity or an individual specifically in 1:1. For example, a breed can present only one size or belong only to sporting type. On the contrast, existential restrictions were created to state that individuals must present at least one instance of a property to be related to a class. For example, some dogs of the breed Great Dane appear to have fatal stomach Bloat Problem, others have heart defects, others hip bone problems or spinal problems. By creating the existential restriction on that, these Genetic Ailments can be related to Great Danes or some to a specific individual that belongs to the class.

Based on all the steps described above in combination with the restrictions, the classes were successfully turned into defined. Each dog breed served as primitive class and the separation was mainly focused on the Characteristics. The idea behind this was to make the defined classes into the Characteristics section, because it seemed more user-friendly to create a list of breeds according to their characteristics and not vice versa.

Therefore, new information was inferred, because all the subclasses of Characteristics superclass were categorizing dog breeds to corresponding categories. For example, in the sizeCharacteristic, after making all subclasses defined classes, the size of each breed was easily referred to the according size class. This process was followed for every class of the Characteristics, creating accuracy and sufficient distinction between them.

In the Figure 5, we can see the result of Luna (individual) which is an Akita Dog, and how it was inferred into the defined classes that it belonged into.

## Queries

Another important aspect of the ontology and the whole project in general, were the Queries that were created using SPARQL. Queries, through Protege, are a powerful way to retrieve and combine information existing in the Ontology. The concept for extracting specific information was sequential, spotting the first URI and then moving towards building the triplets requested, based to what is being searched for.

Some of the queries used are presented below:

A screenshot of a computer

Description automatically generated

Figure 8: Protege SPARQL Query, finding breed classes which are suitable for kids, very intelligent and have low grooming requirements.

This Query is tailored towards retrieving information from classes and not from individuals. The first part starts from retrieving all the Subclasses of Dog breeds and then the ones that have as a suitability for children characteristic either moderate or suitable. Then the second “group” of triplets retrieves the Dog breeds that are only from the category of Brightest Intelligence. The last part of the query retrieves the Dog breeds that belong in the once per week or once in few weeks grooming frequency. In SPARQL all of the aforementioned three parts of the query are connected with the logical AND. This means that the breeds that are retrieved and shown in the lower part of the Figure 8, are satisfying all the conditions. So, the question in natural language here, could be: What are the breeds that are most intelligent, do not have daily grooming needs and at the same time are suitable for children?

A screenshot of a computer program

Description automatically generated

Figure 9: Protege SPARQL Query, finding breed classes which are either working or herding type and at the same time cheap.

Furthermore, the Query shown in Figure 9, is again retrieving information regarding classes. With the same logic that was described above, the first part identifies all the dog breeds that belong to the herding or the working type and then the ones that are characterized as having cheap lifetime cost. The question in natural language here could be: “What will be a cheap and suitable dog for a farmer or someone that lives in the countryside?”

A screenshot of a computer

Description automatically generated

Figure 10:Protege SPARQL Query, finding dog Individuals that have longevity >=8 years and sorting them based on longevity.

Moving on, the Query shown in Figure 10, is retrieving information for the individuals that have been imported into the Ontology. It lists all the individuals that have longevity above 8 years. Here the str command in the selection clause of the query, is used in order to hide the URI that was present in the visual part. Finally, the query displays the name of the aforementioned individuals by ascending order of their longevity.

A screenshot of a computer

Description automatically generated

Figure 11:Protege SPARQL Query, finding dog Individuals, counting the Individuals by size categories and printing how many it did find into each category.

Last but not least, in the query of Figure 11, calculated the size category that each individual belongs to and then counted the result. This was achieved through the use of the group by command together with the count in the select clause. So, the result that was brought back indicated how many individuals were in each category size group.

More queries were implemented and can be found together with their explanations in Appendix 5.

By using Queries in the ontology, a specific target category was created and “fed” into the schema to extract specified properties and object of every instance. As presented above, Queries hold a significant role in extracting and combining attributes. A logical question rising from this section refers to whether they answer to the Open World Assumption.

For example, it is possible that an individual dog, presenting some specific characteristics, are not usually found in the breed. A query extracting medium size and sporting type will result in some entities, but in real-world scenario not all dogs in every breed represent all characteristics similarly. Finally, if the conditions of the query do not match any instance of class or individual, then it will not bring back any result, which means that the knowledge graph does not contain partial information for the specific attributes.

# Conclusion and Future Work

To finalize, the ontology managed successfully to pass the Hermit reasoner test (see Appendix 6). It infers new information about all breeds and individuals contained in the knowledge graph. An interesting idea for future work is to make the dog breed classes defined. In that way, each breed would represent a complete entity with sufficient conditions and new individuals that represent dog instances could automatically be assigned to the respective breed. An interesting addition would be to load more characteristics and expand the depth of the ontology further, starting from the ones that were not used from the original Kaggle dataset. Subcategories evaluating behavioral traits and health care requirements would expand the created knowledge base. In the case of using more additional classes, the tryout of the probe class would be an ideal way to reassure that the structure of the ontology is correct. This could work as an “impostor” class to check if the inferred information is transformed correctly. A fascinating aspect of knowledge graphs refers to data exchange, which could create a deeper and more meaningful analysis. For example, combining our dog ontology with an ontology based on cats could promote the integration and sharing of data based on both animals across different bases. In that case the probe class would again seem appropriate, since it is recommended for data integration. Overall, this project provided a comprehensive ontology based on dog breeds, extracting key-characteristics to categorize dogs and shape a useful, high in variety and cute knowledge graph.

# References

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform resource identifier (URI): Generic syntax* (No. rfc3986).

Berners-Lee, T., Hendler, J. and Lassila, O., 2001. The Semantic Web. Scientific American, May 2001.

*Dog breeds ranked 🐕🐾*. (2024, May 18). Kaggle. <https://www.kaggle.com/datasets/jainaru/dog-breeds-ranking-best-to-worst>

*Dog*. (n.d.). DBpedia. <https://dbpedia.org/page/Dog>

Kejriwal, M., 2022. Knowledge Graphs: A Practical Review of the Research Landscape. Information, 13(4), p.161. Available at: <https://doi.org/10.3390/info13040161>

Kück, G., Tim Berners-Lee's Semantic Web. Leaf Wireless (Pty) Ltd. South African Journal of Information Management, May 2004.

Libkin, L., & Sirangelo, C. (2009). Open and closed world assumptions in data exchange. In Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009. Retrieved from DBLP

Schmidt, D., Trojahn, C., & Viera, R. (2018). Matching BFO, DOLCE, GFO and SUMO: An evaluation of OAEI 2018 matching systems. In Proceedings of the OAEI 2018 Workshop.

Zou, X., 2020. A Survey on Application of Knowledge Graph. Journal of Physics: Conference Series, 1487, p.012016. Available at: <https://doi.org/10.1088/1742-6596/1487/1/012016>

# Appendices

## Appendix 1

All the subclasses of Dog class, until depth three.

* Dog
* Affenpinscher
* Akita
* AlaskanMalamute
* AustralianCattleDog
* Basenji
* Beagle
* Belgian Tervuren
* BerneseMountainDog
* BichonFrise
* BorderCollie
* Borzoi
* BouvierdesFlandres
* Boxer
* Briard
* Brittany
* BrusselsGriffon
* Bulldog
  + ClassicBulldog
  + FrenchBulldog
* Chihuahua
* ChowChow
* Dalmatian
* DobermanPinscher
* GreatDane
* Hound
  + AfghanHound
  + BassetHound
  + Bloodhound
  + Dachshund
  + Greyhound
  + IrishWolfhound
  + ItalianGreyhound
  + PharaohHound
* LhasaApso
* Maltese
* Mastiff
  + BullMastiff
  + ClassicMastiff
* Newfoundland
* Papillon
* Pekingese
* PembrokeWelshCorgi
* Pointer
  + ClassicPointer
  + GermanShorthairedPointer
* Pomeranian
* Poodle
* Pug
* Retriever
  + ChesapeakeBayRetriever
  + FlatCoatedRetriever
  + GoldenRetriever
  + LabradorRetriever
* RhodesianRidgeback
* Rottweiler
* SaintBernard
* Saluki
* Samoyed
* Schnauzer
  + GiantSchnauzer
  + MiniatureSchnauzer
* Setter
  + EnglishSetter
  + GordonSetter
  + IrishSetter
* Sheepdog
  + OldEnglishSheepdog
  + ShetlandSheepdog
* Shepherd
  + AustralianSepherd
  + GermanShepherd
* ShihTzu
* SiberianHusky
* Spaniel
  + CavalierKingCharlesSpaniel
  + ClumberSpaniel
  + CockerSpaniel
  + EnglishCockerSpaniel
  + EnglishSpringerSpaniel
  + EnglishToySpaniel
  + TibetanSpaniel
  + WelshSpringerSpaniel
* Terrier
  + AustralianTerrier
  + BedlingtonTerrier
  + BorderTerrier
  + BullTerrier
  + CairnTerrier
  + DandleDinmontTerrier
  + KerryBlueTerrier
  + NorfolkTerrier
  + ScottishTerrier
  + StaffordshireBullTerrier
  + TibetanTerrier
  + WestHighlandTerrier
  + YorkshireTerrier
* Whippet

## Appendix 2

A diagram of different types of characters

Description automatically generated

Figure 12:GraphDB visual, showing the Characteristics class with all its subclasses.

|  |  |
| --- | --- |
|  | Characteristics |
| Genetic ailments | Health problems caused by abnormalities in the genome |
| Grooming frequency | How often a breed needs grooming |
| Intelligence | The level of the mental activity for each breed |
| Lifetime Cost | Estimation of the financial cost for each breed |
| Longevity | The life expectancy for each breed |
| Size | How physically each breed will develop |
| Suitability for Children | Whether a breed is safe for children |
| Type | Different activities each breed support |

Table 1:Subclasses of the Characteristics class, with explanations.

## Appendix 3

Some of the columns of the original dataset contained non integer numbers with a wide range. Thus, to avoid confusion and mis categorization, we turned everything into integers and scales were created on specific columns. For example, the Lifetime Cost was set to: Cheap: 13.000-18.000€, Medium: 18.001-23.000€, Expensive: 23.001-27.000€.

|  |  |
| --- | --- |
| Cheap | 13.000-18.000€ |
| Medium | 18.001-23.000€ |
| Epxensive | 23.001-27.000€ |

Table 2: LifetimeCost Data Property Ranges

Moreover, the data property hasLongevity was used to define the lifespan that individual dogs have. The subproperties hasLongevityLong, hasLongevityMedium and hasLongevityShort specified more accurately the longevity of each breed. A scale to better handle the range of the years was produced as following:

|  |  |
| --- | --- |
| Long | 14-17 years |
| Medium | 10-13 years |
| Short | 6-9 years |

Table 3: Longevity Data Property Ranges

That way, numerical values were transformed to categorical scaling. The Longevity of each breed was in a numerical format, so the scaling was set to: Long: 14-17 years, Medium: 10-13 years, Short: 6-9 years.

## Appendix 4

A screenshot of a computer

Description automatically generated

Figure 13:Rule in JSON format, for importing Individuals into Protege.

Here <https://jsonformatter.org/json-viewer> was used to “visualize” the JSON rule, that was used and saved in Protege Ontology, for importing the individuals.

And in Figure 14 we can see the visual representation of the individuals.

A group of names in circles

Description automatically generated

Figure 14:GraphDB visual, showing all the imported individuals in the Ontology.

## Appendix 5

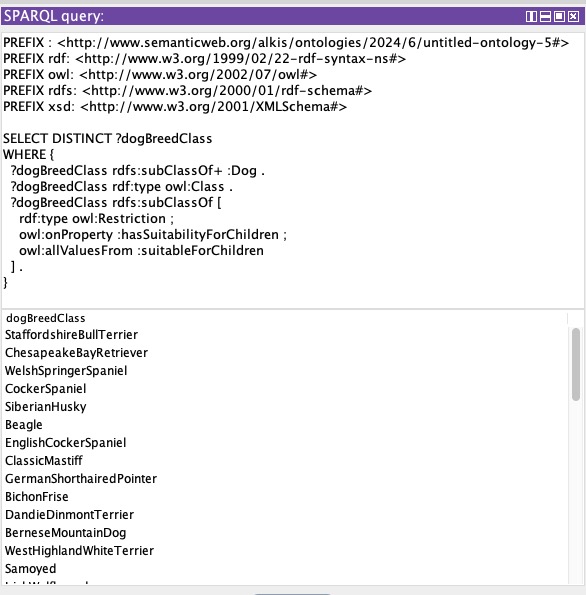


Figure 15: Protege SPARQL Query, finding all Dog Breeds Classes that are suitable for Children.

The Query shown in Figure 15, finds all the Subclasses of Dog and then searches for the property suitable for children and brings back all the breeds that are suitable for children.



Figure 16:Protege SPARQL Query, finding all Dog Breeds with all thire Longevity types, binding the type as words, and showing the 10 of them in alphabetical longevity order.

The Query shown in Figure 16, finds all the Subclasses of Dog and then searches for the property of long longevity. Then in the second part (UNION), it combines the results with the subclasses of dogs that have short longevity and in the third part combining both of them with the dogs that have medium longevity. In each of the union parts, each category is assigned a specific visual value (with the use of BIND command) and then the results are brought back in alphabetical order based on the binded values and it brings back only 10 of them (LIMIT).

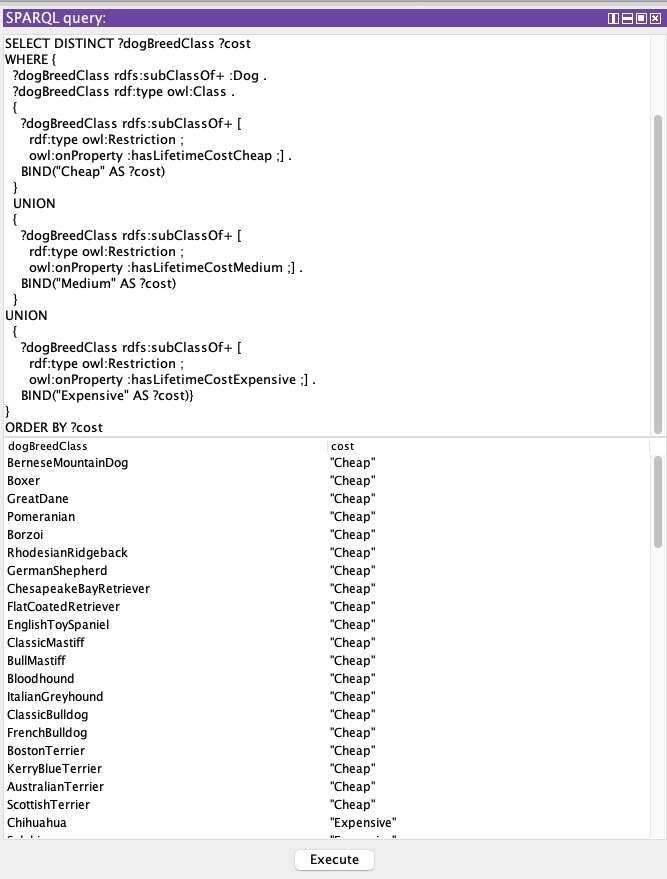


Figure 17: Protege SPARQL Query, finding all Dog Breeds with all their Costs categories, binding the cost category as words, and showing them in cost alphabetical order.

The Query shown in Figure 17, does exactly the same with the above query but for cost instead of longevity.



Figure 18:Protege SPARQL Query, finding all the Individuals together with the Dog Breed subclass that they belong to.

The Query shown in Figure 18, deals with individuals. It brings back all the individuals names in the first column and in the second column the Dog class that they belong to.

## Appendix 6

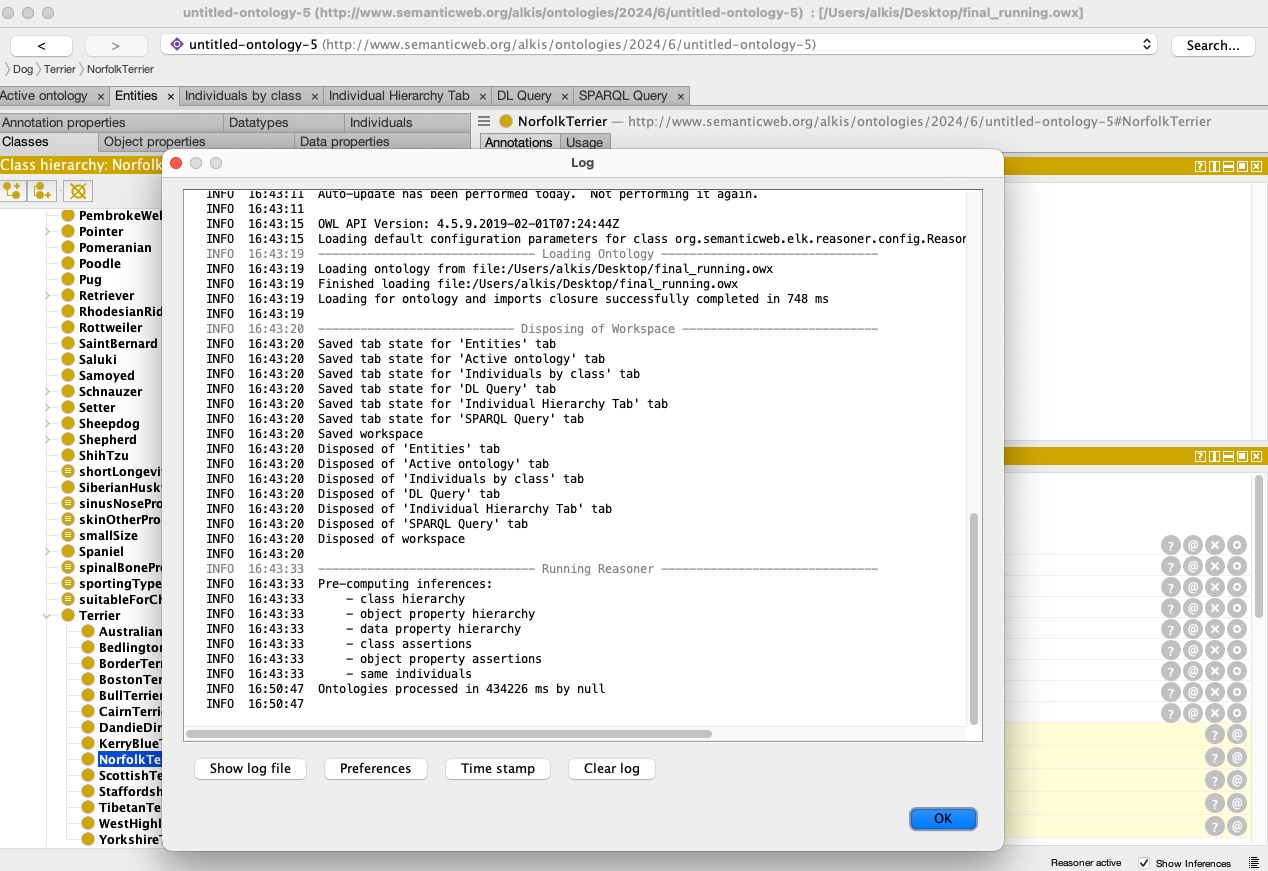


Figure 19: Protege Screenshot, of the full Ontology after successfull running of the Hermit Reasoner.

As shown in the Figure 19, the delivered ontology file was able to run the Hermit 1.3.8.413 Reasoner successfully without indicating any errors.

## Appendix 7

To enhance the understanding of the dataset, some visuals were generated using the seaborn library in python.

A group of graphs showing different sizes of dog breeds

Description automatically generated

Figure 20:Python Seaborn visual, showing relationship of characteristics with size.

The scatterplot represents the relationship of each characteristic by pairing them, having set the size of dogs as the ‘hue’ parameter.

A graph of a number of years

Description automatically generated

Figure 21: Python visual, showing the distribution of Longevity.

A graph with dots and numbers

Description automatically generated

Figure 22:Python visual, showing each breed according to the defined longevity.