**Web Pokémon Poked**

**Introduction:**

The purpose of this project was to explore the semantic relationships that occur within the Pokémon videogame universe. After several avenues of research it was determined that there is little to no publicly available data in an RDF format that can easily be retrieved. There are small data sets, like the one provided at http: //datahub.io/dataset/data-incubator-pokedex, but none that completely cover the Pokémon universe. There is also an issue that these sources are not governed by any form of standardized ontology. The Pokémon website, Veekun, located at http://veekun.com provides access to a complete, updated, and publicly available SQLite database. This database provided a substantial amount of information about the Pokémon gaming universe. The goal was to take the data provided within this database and then port it into a turtle RDF format that could be used by Jena to make inferences. Due to the sheer size of the dataset and lack of additional team member support, the dataset was trimmed down to focus strictly on the data relating to Pokémon within the red/blue version of the Pokémon game. Ontology was also constructed to relate this dataset together semantically.

**Ontology Owl Classes:**

* + Item

This was a class to represent all items that could be utilized by a Pokémon. These items were pruned down though to focus on two instances of items, HM’s and TM’s. These items are designed to teach moves to Pokémon. Class instances of this are related to class instances of Move to establish a relationship that Pokémon can be taught moves by items.

* + Species

This class is supposed to be the representation of the actual Pokémon. Class instances of this are mostly related to primitive values that provide specific details about the Pokémon, such as its height and weight. This class is also related to all of the other classes within this ontology in the following ways:

* + - Move

This is a representation of what attacks or status buffs Pokémon can or will learn naturally to use against other Pokémon in combat. This class instance is also related to several primitive values that define its accuracy, power, description, and the number of times the movie can be used.

* + - Family

This is a representation the type of classification that a Pokémon is. This is synonymous to the species families that we use, like primate. The specific family a species is assigned to have very powerful implications within the Pokémon universe. The family that a species is assigned to will dictate how much damage it does to other Pokémon species that have a different family type. Some family relationships will allow for double the amount of damage done to another Pokémon. In other instances, damage can be halved or nullified all together.

* + - Habitat

This defines where specific Pokémon can be found within the realm of the Pokémon universe. Within the context of this dataset it has little significance other than informational purposes, but has stronger implications in other generations. These implications can go to define the temperament of a Pokémon and specific gender within later versions of the Pokémon game.

* + - Rate of Growth

This defines how fast a specific species of Pokémon will grow. This is rather important because it defines the amount of time and battles it will take for a Pokémon to level up. This affects such things as when that Pokémon will learn a move, evolve, and allow the player to access certain areas of the game.

* + - Evolution

This is a critical milestone that Pokémon can meet. It provides a stipulation that allows for species instances to essentially become another, completely different species instance. For example, the Pokémon species Charmeleon undergoes a level evolution at level 36. This can allow that Pokémon instance related to Charmeleon to become assigned the species of Charizard when the Pokémon reaches level 36. It will then take on all of the benefits and burdens associated with the Charizard species, losing completely its identity as a Charmeleon. This milestone occurs under one of three conditions. Either the Pokémon is exposed to a specific stone, reaches a level, or is traded to another player.

**Implementation & Results:**

The original dataset was stored within a SQLite database in a relational format. The database consists of roughly 110 tables with about 5 million records. The original intent was to implement a full scale implementation of this database in RDF, but due to this being such a diverse dataset that is in several different languages, a sampling was used instead. The goal of the project was to provide a similar experience as found when using a pokedex found in the Gameboy games. To provide this experience and still work with a constrained dataset, First Generation games Red/Blue were the targeted problem domain. The data was extracted using SQLite Manager Plugin for Mozilla Firefox to pull the data into a csv format. Using the OpenCSV java library, the csv files were then parsed into triples using RDF turtle format.

Once the triples were in an RDF format, an ontology was created that defined class types, evolution transitive relationships, property domain and ranges, family/move sub-property hierarchy, etc. This ontology is embedded with the final dataset at the bottom of all of the triples. Using this ontology, data triples, and Jena Reasoner java libraries an inference graph was able to be created. The following three java classes were used to construct and manipulate the Jena Reasoner libraries and RDF dataset.

* jena\_graph.java

The goal of this class was to provide a simplified wrapper for the web\_builder.java class to interact with the Jena graph. The logic for doing this was to separate the graph portion of the implementation and the web building portion to simplify troubleshooting with graph specific issues. This class provides support for dynamic select and ask queries, applying OWL ontology specifications to an existing graph, constructing the graph, and releasing the resources used by the graph database.

* web\_builder.java

This class is the workhorse of the Pokedex application. The class begins by taking an incoming faceted search string from the web interface and then dereferencing it into separate tokens. Taking each token it interpreted a specific leading symbol and created an appropriate SPARQL triple to represent the meaning of the token. After all of the tokens were interpreted, their respective SPARQL triple representations are taken and appended to the end of another SPARQL query that retrieves all of the id values associated with Pokémon.

Taking the previous query result set, the web\_builder.java class loops through each value and retrieves the specific pokedex information associated with each id in the result set. The class then constructs html tags to format the returned pokedex information about a Pokémon. Because of how Jena returns SPARQL query results for URI and integer primitive values, they needed to be dereferenced to remove extra markup to get the desired values. URI and integer results are ran through a function that performs this task and then returns a separated string representation of these values. After all id values have been exhausted from the original result set, the program appends any closing tags to the html file and then passes priority back to the PokedexServlet.java class.

* PokedexServlet.java

This class was merely designed to provide web support for the web\_builder.java class. Its only function is pass the search query from the html interface to be dereferenced by web\_builder.java, release the resources used by web\_builder.java when it’s done building the webpage, and then redirect the browser to the newly constructed html file produced by the web\_builder.java class. The purpose for doing this was merely based on personal preference. It was easier to write the html line by line to a separate file then attempting to build an html page in memory.

To provide web support for the search interface and query results, Apache Tomcat was used as the project’s webserver. The PokedexServlet.java was then deployed onto the Tomcat server which was housed on a personal machine. The Tomcat server then provided a web interface to interact with the dataset through queries which was accessible through a web browser referencing the local host address on the personal machine.