League of Legends Ontology

By Justin Hu and Shubham Vij

Domain

Both of us have a strong interested in a game that has taken the world by the storm, League of Legends. We saw that the game, although has various wikis and resources to find game information, it does not have a smart Question Answer system that can allow a new player to learn about the game and an experienced player an edge over others by learning interesting quirks that others might not know. Essentially the goal for creating this ontology was to provide a tool for players to better understand the game no matter the play level. Upon creating the ontology we quickly realized that this task is highly complex, as the overall game domain is huge! Thus, for this assignment for the convenience of time we limit the domain to a set of characters and their interactions with some ingame properties and modifiers. A user may ask questions such as what skills does a character have, or what character has specific abilities, or what other characters are similar in playstyle.

Developing the Ontology

Our first steps included mapping out the purpose of the ontology, as discussed above. Secondly we though about the domain and what questions were that we wanted to answer.

```
    Determine the domain and scope of the ontology
```

We came up with a list from which below is an excerpt:

- What champions have specific abilities?
- What is a champion that can be played as a tank?
- How do I counter this champion?
- How do items modify champions and other characters?
- Who is ____ and what abilities, buffs and properties does it have?
- How difficult is it to master a certain champion?
- What items go well with a given champion?
- ...

Above is just a short representation of 60+ compotency questions that we derived.

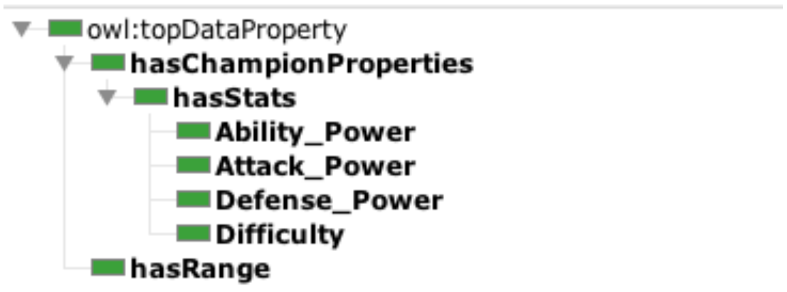
```
    Define the classes and the class hierarchy
```

Next, we started mapping out the higher level classes. We broke down the domain to a list of 8 high level classes. Thereafter, we mapped the class hiearchy we though would work well in a top down fashion. After that we started to map out some of the object relations between the classes. We soon realized that this method was not working as our class relation logic was complex, did not work, and was quite confusing. We decided then to instead map out what the relations we wanted and then mapped out a new set of class hiearchy. This actually took multiple iterations before we had an adequate handle on a logical class hierchy and inter-class relations. This process worked, but we could have greatly improved the efficiency of this process by outlining some logical tests first and then laying out the class structures. Roughly spending a couple days on this we realized that the domain we were trying to tackle was huge! Thus, we cut down the 300+ classes we outlined to just 140 classes - yes still huge, but we believed we could tackle it, up until we started to map out the individuals. Most of these classes were the union of defined and complex classes.

```
Enumerate important terms in the ontology
and, Define the properties of classes
```

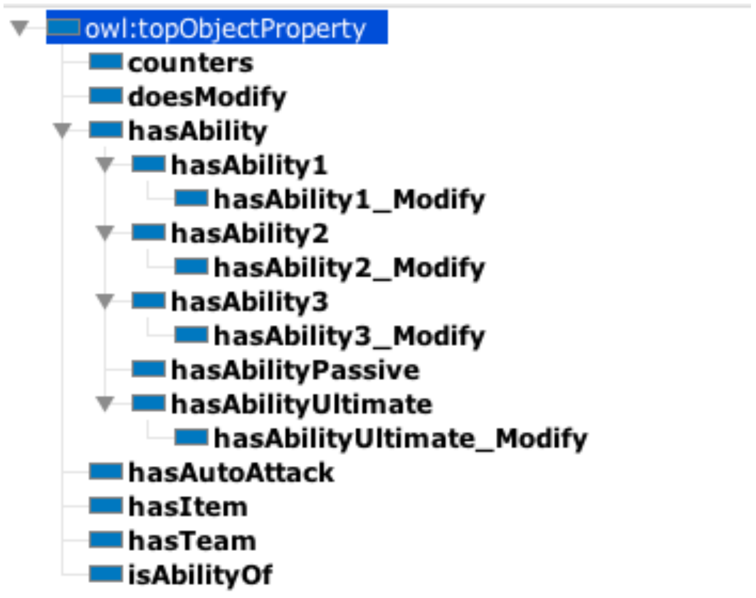
Next we started to define the properties of each class, again came to a realization that we needed to further limit the scope of these object properties. Thus, we decided to focus on just one subsegment of our mapped classes for our object properties. We decided to break down the object properties into basic attributes that a character might have, such as health, mana, auto attack type, types of abilities, etc. We soon came to realize that some of these are better suited to be data properties instead, such as health, mana, attack power and so on; which are better represented as a bounded integer rather than seperate classes. In the end we found that the following object and data properties worked best for our scenario - some of whom are inverse object properties and some are bounded data properties.

Data Properties



```
owl:topDataProperty
    hasChampionProperties
        hasStats
            Ability_Power
            Attack_Power
            Defense_Power
            Difficulty
    hasRange
```

Object Properties



- owl:topObjectProperty
  - counters
  - doesModify
  - hasAbility
    - hasAbility1
      - hasAbility1_Modify
    - hasAbility2
      - hasAbility2_Modify
    - hasAbility3
      - hasAbility3_Modify
    - hasAbilityPassive
    - hasAbilityUltimate
      - hasAbilityUltimate_Modify
  - hasAutoAttack
  - hasItem
  - hasTeam
  - isAbilityOf

```
    Create instances
    and, Re-Define the classes and the class hierarchy
```

Now came the decision on whether on not to represent champions in the game as individuals or classes. There were benifits and cons to both approaches. But,one con that forced us to adapt the individual scheme is having them as classes dissallowed us to use the benifit of defining just one changable data property of Health, Mana, etc. for each champion. Now, was time to map out all of the individuals. After mapping around 77 individuals of both champions and abilities, yet another realization was made as to the structure of classes. Through this process it was discovered that some of our needed questions could actually not be answered and needed a class and relationship restructuring. Our final class hiearchy was a result of using a bototm up approach - accounting the current top down class structure we had along side the individuals that were mapped out. This allowed us to create a much cleaner class hiearchy with only 114 classes. Some of the object property connections we had moved to parent classes, eliminating the need for some of the classes all together. Other times, we were able to generate new classes to encapsulate a broader number of other same level classes. Through multiple iterations in this process we were able to finalize the class structure.

```
    Ensuring that the class hierarchy is correct
```

At this time, we tested the class hierarchy by running manual tests using DL Query and used OntoGraph to visually walk through the hierchy to ensure that everything was in order and correcting any mistakes that we came accross.

```
    Defining properties—more details
```

Having made the base ontology and class properties, we finished the ontology by adding some missing connections and data properties.