A Semantic Web-based Approach to Achieve Pokédex

GAO Zhenyu[1][2633314] and LIU Rui[2][2644856]

[1] VU, the Netherlands vu.gaozhy@gmail.com
[2] VU, the Netherlands liurui_better@163.com

Abstract. In this project, we implemented a semantic web-based approach to achieve a knowledge representation system related to "Pokémon". We linked different sources including several external datasets, designed the ontology and constructed the knowledge graph using the knowledge and tools related to knowledge representation for the web. Our system has a clear structure with which we can easily search information of the "Pokémon" and also make several advanced queries.

Keywords: Knowledge representation · Knowledge graph · Linked Data · Semantic Web.

1 Introduction

As the highest-grossing multimedia franchise in the world, the Pocket Monsters (Pokémon, owned by The Pokémon Company, Nintendo, Game Freak, and Creatures Inc.) has achieved great successes in several areas, including video games, mobile games, animes & movies, etc. In addition to these successes, a remarkably fact is that through the 24-year history of Pokémon, several notably datasets have been constructed and constantly updated. Among these datasets, the most noteworthy one is the one based on the video games, which is the origin of the whole Pokémon universe.

The dataset of Pokémon video games is a large, well-organized dataset with multiple stats, covering from the basic species information to detailed battle information. There are hundreds of websites that can provide different kind of information regarding Pokémon. However, despite the abundant data sources of Pokémon on the World Wide Web, very few Semantic Web-based sources can be found at the moment. Given this phenomenon, in this paper we aim to use the linked data approach create a Semantic Web of the Pokdex (i.e. a detailed knowledge representation of Pokémon on the web), which allows the user to not only perform a variety of queries regarding the video games, but also achieve a basic simulation of the Pokémon battle.

The Semantic Web is applied as the extension of current websites like those in World Wide Web, aims to enhance the flexibility and usability, and improve the interaction between agent and computer[1] [2]. These goals are achieved by a sequence of approaches, including ontology engineering, using uniform resource

identifiers(URIs), external database linking, the Resource Description Framework (RDF) language and SPARQL language [1][3][2][4][5].

Given the advantages of Semantic Web and our goal narrated above, the rest of the paper is constructed as follows: literatures and works related to the research question are presented in Section 2. In Section 3, the approaches of achieve our goal are discussed in detail. The experimental outcomes are showed in Section 4, including the result of visualization, representation in Triply, and the simulation. Finally, in Section 5, the conclusions, limitations and future works are discussed.

2 Literature Review & Related Work

Accompanied by the development and popularity of Pokmon video games, several datasets have constructed based on the World Wide Web. For instance, serebii.net[3] is a English Pokmon dataset that can achieve basic queries based on the generation and type of the Pokmon. Baubapedia[4] is a dataset with 7 languages that contains elaborate information encompassing video games, card games, animes and movies. As for the battle information, we can find it through the official website Pokmon Global Link[5].

The idea of Semantic Web was put forward by Tim Berners-Lee and his colleague[1], from which several concepts and approaches are persistently inspiring the researches till now. [6] has proposed 4 rules to instruct linked data, including: (1) give object names with URIs, (2) the URIs should be searchable for users, (3) using RDF or SPARQL to give useful information regarding URIs, (4) URIs should be linked with each other. Related to our purpose, [7] have proposed a framework to build games, using linked data refinement.

3 Approach

3.1 Dataset

Among several websites concerning Pokémon, we chose the dataset from serebii.net[3], which covers a very detailed information we want. From this database we get the `.csv` file, based on which we designed our ontology and established the based RDF construction.

Also, integrating one source with other datasets is important, for it can cut the redundant data by building relations with present datasets [4]. In our study, we linked the dataset we constructed to DBpedia, the detailed database with Triple Patterns, so that we could manipulate the linking work by SPARQL, resulting a new interface for users to access different information[8].

[3] `https://www.serebii.net`
[4] `https://bulbapedia.bulbagarden.net`
[5] `http://www.pokemon-gl.com`

3.2 Ontology Design

We designed the ontology based on the dataset (see Figure 1). As a dataset of Pokémon, the ontology is a relatively simple one, for the core of the ontology is the "Pokemon", each Pokémon has its unique abilities and attributes.

Another important object in the dataset is the "type" (of a Pokémon), which is related to the weakness and resistance of a Pokémon, can these information can be queried individually (i.e. without indicating a specific Pokemon).

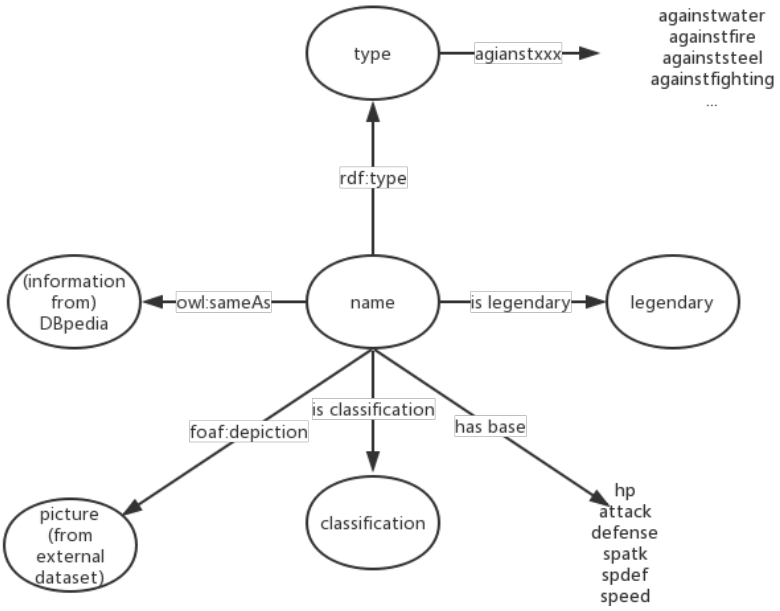

Fig. 1. The ontology of Pokémon.

3.3 Knowledge Graph Construction

We used GraphDB for knowledge graph construction, visualization and data conversion. Based on the ontology and data sources, we did the construction work in two parts: "Pokémon" part and "Type" part with OntoRefine using SPARQL language. Besides the relations defined by ourselves, we used rdfs, foaf, and owl to present our relationship. (the detailed code can be seen in the appendix)

3.4 External Dataset Linking

To enrich our dataset, we linked several external datasets. We used `owl:sameAs` in order to link to the DBpedia website and an existing demo knowledge graph[6] in Triply for each Pokémon. We implement this by first using SPARQL query to get all the triples and save it in a `.ttl` file, and then using Python code to merge it with our exsiting `.nt` file (all these code are inclueded in the appendix).

We also linked to serebii.net[3] and pokemonshowdown.com to get pictures with the format of `.jpg` (static images) and `.gif` (moving images).

4 Experiments and Results

4.1 Visualization

Using GraphDB, we can realize the visualization of the relationship of our dataset. As we can see in the pictures (see Figure 2), when we search for a Pokémon called Incineroar, the result will show the related information, including number, species, generation and type. When we search for a ceartain type like "Electric", we can see the electric Pokémon (see the red circles in Figure 3), and the number of types which can do 0.5, 1.0 or 2.0 times the damage to Electric Pokémon (see the green circles in Figure 3).

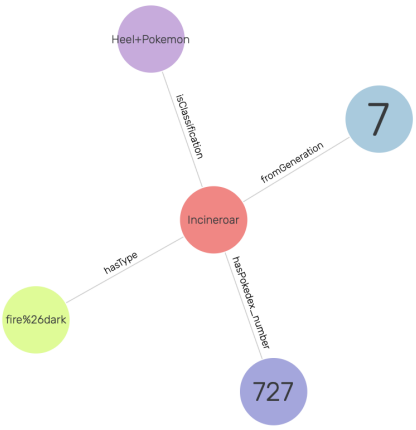

Fig. 2. Visualization of "Incineroar"

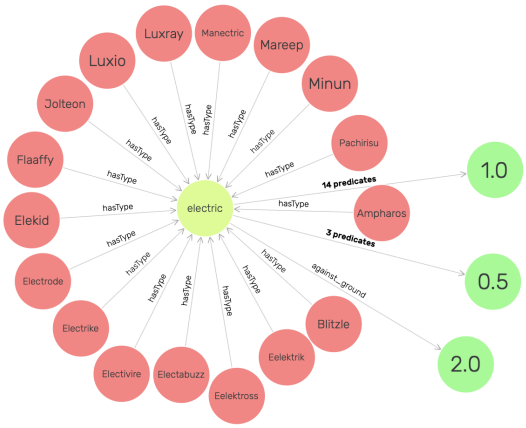

Fig. 3. Visualization of "electric"

In Triply, we can also see these relationships. For example, if our search target is "Pikachu" and we type it in the browser of Triply[7], we can find all the information related to "Pikachu" (see Figure 4).

[6] `https://demo.triply.cc/academy/pokemon`

[7] `https://krr.triply.cc/lafengzhenyu/datasetgifquery/browser?resource=http%3A%2F%2Fexample.com%2Fpokemon%2FPikachu`

Futhermore, we know that "Pikachu" is an "electric" Pokémon because it has type of "electric", and then we can find all the Pokémons in our knowledge graph which belong to "Electric" Pokémons[8] (see Figure 5).

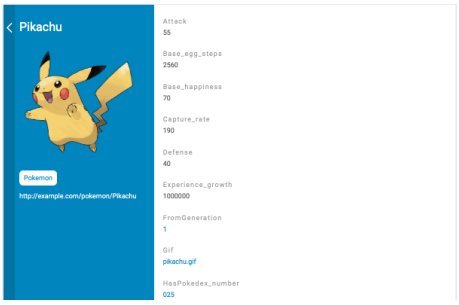

Fig. 4. "Pikachu" in the Browser

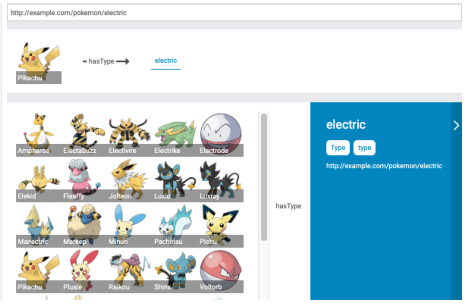

Fig. 5. "Electric" Pokemon in the Browser

4.2 Queries using SPARQL

We implement two kind of queries using SPARQL. The one is called "Battle System", and the other is "Enemy System".

Battle system Battles are usual and important for Pokémons, therefore we design a battle simulation system for them. The results of battles are based on

Fig. 6. A 1V1 battle prediction

[8] https://krr.triply.cc/lafengzhenyu/datasetgifquery/browser?resource=http%3A%2F%2Fexample.com%2Fpokemon%2Felectric&focus=backward

base-attack, base-defense and HP values of the pokemons. We use a relatively simple equation to predict it.

$$HP\_after = HP\_before + prob \times (defense - attack\_enemy)$$

where $prob \in (0, 1)$ is a random number which means the multiple of those base values.

We define that the pokemon(in 1V1 battle) or the combination(in 2V2 battle) has higher HP value will win the battle.

For exmaple, if we choose "MACHAMP" and "BLASTOSE" to join the battle, after one round, the HP value of "MACHAMP" is higher and our system predicts that he will win the battle. (see Figure 6).

Similarly, when we choose "SUICUNE" and "PICHU" versus "DRAFONAIR" and "AMBIPOM" which are a 2V2 battle, our system predicts the latter combination will win (see Figure 7).

Fig. 7. A 2V2 battle prediction

Find Enemy system Each pokemon has it own type, and Pokémon types dictate whether one Pokémon is either strong or weak - super effective or not very effective - against another, dealing additional or reduced damage as a result - and receiving additional or reduced damage - as a result. For example, type 'Fighting' is strong against type 'Normal', 'Rock' and 'Ice', but weak against type 'Flying', 'Poison' and 'Psychic'. Therefore, we define the enemies of a Pokémon are the the Pokémons whose types are strong against him.

If we want to find the enemies of "Pikachu" whose type is "Electric", the system will give us several samples of "Ground" Pokémon (see Figure 8) because type 'Ground' is strong against type 'Electric' ("2" means leading to double damages).

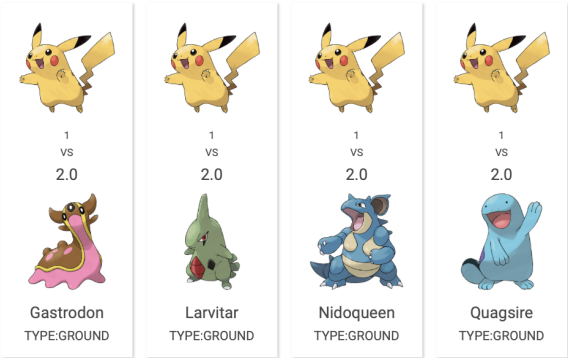

Fig. 8. the enemies of "Pikachu"

As an another example, the enemies of "Leavanny" have more kind of types because it has both types of "Bug" and "Grass" (see Figure 9). We can see that "Wingull" will cause 4 times damages to "Leavanny" and the reason is "Flying" is both strong against type "Bug" and "Grass".

Fig. 9. the enemies of "Leavanny"

4.3 Evaluation

Through Visualization we can clearly understand the knowledge graph of our Pokémon system. And The two queries can make use of the architecture of our ontology and knowledge graph to inference and predict something new. Combined with `HTML`, the system can give us a relatively clear visual result.

5 Conclusion

In this paper, we aim to construct a database based on the theories of Semantic Web. To realize this goal, we did a series of approach: choosing datasets, designing ontology, constructing knowledge graph, converting data, linking external datasets and implementing several useful queries.

Through these approaches we have achieved a "Pokédex" with most of the information as well as some query-based functions, yet a lot should be done for improvement. For example, the ontology in our case seems limited, in the future we can look into more aspects, extend the ontology by seeking correlations between concepts, instances or properties[9]. For the query part, we can make more interesting and useful simulations[10], and we can also improve our the interface by using more complicated `SPARQL` and `HTML` sentences.

Also, some relations in the knowledge graph may have a better definition than the existing ones, with some update of these relations we may able to perform more interesting functions.

References

1. T. Berners-Lee, J. Hendler, O. Lassila, *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.
2. X. Ma, Y. Chen, H. Wang, J. Guang Zheng, L. Fu, P. West, J. Erickson, and P. Fox, *Data visualization in the Semantic Web*, pp. 149–167. 08 2015.
3. C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," in *Semantic services, interoperability and web applications: emerging concepts*, pp. 205–227, IGI Global, 2011.
4. N. Petersen, L. Halilaj, I. Grangel-González, S. Lohmann, C. Lange, and S. Auer, "Realizing an rdf-based information model for a manufacturing company–a case study," in *International Semantic Web Conference*, pp. 350–366, Springer, 2017.
5. N. Shadbolt, T. Berners-Lee, and W. Hall, "The semantic web revisited," *IEEE intelligent systems*, vol. 21, no. 3, pp. 96–101, 2006.
6. T. Berners-Lee, "Linked data." `https://www.w3.org/DesignIssues/LinkedData.html`. Last Accessed June 18, 2019.
7. G. R. Calegari, A. Fiano, and I. Celino, "A framework to build games with a purpose for linked data refinement," in *International Semantic Web Conference*, pp. 154–169, Springer, 2018.
8. O. Hartig, I. Letter, and J. Pérez, "A formal framework for comparing linked data fragments," in *International semantic web conference*, pp. 364–382, Springer, 2017.
9. K. Gunaratna, S. Lalithsena, and A. Sheth, "Alignment and dataset identification of linked data in semantic web," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 2, pp. 139–151, 2014.
10. N. Richerzhagen, B. Richerzhagen, D. Stingl, and R. Steinmetz, "The human factor: A simulation environment for networked mobile social applications," in *2017 International Conference on Networked Systems (NetSys)*, pp. 1–8, IEEE, 2017.

A Datasets Sources

The dataset we have chosen is available at `https://github.com/lafengxiaoyu/KRoW/tree/master/pokemon%20databases`. The data is converted by GraphDB, in which after construction we convert RDF to N-Triples.

B Instance Linking

B.1 Battle System

1V1 Battle: `https://krr.triply.cc/lafengzhenyu/datasetgifquery/queries/Battle1v1/2`

2V2 Battle: `https://krr.triply.cc/lafengzhenyu/datasetgifquery/queries/battle2v2/2`

B.2 Find Enemy System

`https://krr.triply.cc/lafengzhenyu/datasetgifquery/queries/findenemy1/2` `https://krr.triply.cc/lafengzhenyu/datasetgifquery/queries/findenemymany/1`

C Codes

C.1 knowledge graph construction

We use SPARQL and GraphDB to construct the knowledge graph based on our ontology.

We have another SPARQL code which is available in our github `https://github.com/lafengxiaoyu/KRoW`

```
#constructing Pokemon graphs with index=name
PREFIX mdb: <http://example.com/pokemon/>
PREFIX spif: <http://spinrdf.org/spif#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
#relating basic informations, linking pictures and DBpedias
#using INSERT when adding to graph
CONSTRUCT{
 ?myRowId a mdb:Pokemon ;
  mdb:name ?name ;
  mdb:japanese_name ?japanese_name ;
  owl:sameAs ?DbpediaIRI;
  mdb:gif ?PictureBattle;
  foaf:depiction ?Picture;
```

```
  mdb:isClassification ?classificationIRI ;
  mdb:hasPokedex_number ?pokedex_numberIRI ;
  mdb:fromGeneration ?generationIRI ;
  mdb:hasType ?typeIRI ;
  mdb:abilities ?abilitiesIRI ;
  mdb:height_m ?height_m ;
  mdb:weight_kg ?weight_kg ;
  mdb:experience_growth ?experience_growth ;
  mdb:base_happiness ?base_happiness ;
  mdb:base_egg_steps ?base_egg_steps ;
  mdb:capture_rate ?capture_rate ;
  mdb:percentage_male ?percentage_male ;
  mdb:is_legendary ?is_legendary ;
  mdb:attack ?attack;
  mdb:defense ?defense;
  mdb:hp ?hp;
  mdb:sp_attack ?sp_attack;
  mdb:sp_defense ?sp_defense;
  mdb:speed ?speed.

 ?classificationIRI a mdb:classification;
  foaf:name ?classification.
 ?pokedex_numberIRI a mdb:pokedex_number;
  rdfs:value ?pokedex_number.
 ?generationIRI a mdb:generation;
  rdfs:value ?generation .
 ?typeIRI a mdb:type;
  foaf:name ?type .
 ?abilitiesIRI a mdb:abilities;
  foaf:name ?abilities .
 ?DbpediaIRI a mdb:Dbpedia;
  foaf:name ?Dbpedia.
 ?PictureBattle a mdb:PictureBattle.
 ?Picture a mdb:Picture.
} WHERE {
          ?row a mdb:Pokemon ;
           mdb:name ?name ;
           mdb:Dbpedia ?Dbpedia;
           mdb:japanese_name ?japanese_name ;
           mdb:classification ?classification ;
           mdb:pokedex_number ?pokedex_number ;
           mdb:generation ?generation ;
           mdb:type ?type ;
           mdb:abilities ?abilities ;
           mdb:height_m ?height_m ;
```

```
            mdb:weight_kg ?weight_kg ;
            mdb:experience_growth ?experience_growth ;
            mdb:base_happiness ?base_happiness ;
            mdb:base_egg_steps ?base_egg_steps ;
            mdb:capture_rate ?capture_rate ;
            mdb:is_legendary ?is_legendary ;
            mdb:PictureBattle ?PictureBattle;
            mdb:Picture ?Picture;
            mdb:attack ?attack;
            mdb:defense ?defense;
            mdb:hp ?hp;
            mdb:sp_attack ?sp_attack;
            mdb:sp_defense ?sp_defense;
            mdb:speed ?speed.
            OPTIONAL{?row mdb:percentage_male ?percentage_male}
          BIND(IRI(concat("http://example.com/pokemon/", spif:encodeURL(?name)))
          BIND(IRI(concat("http://example.com/pokemon/", spif:encodeURL(?classifi
          BIND(IRI(concat("http://example.com/pokemon/", spif:encodeURL(?pokedex_
          BIND(IRI(concat("http://example.com/pokemon/", spif:encodeURL(?generati
          BIND(IRI(concat("http://example.com/pokemon/", spif:encodeURL(?type)))
          BIND(IRI(concat("http://example.com/pokemon/", spif:encodeURL(?Dbpedia)

        }
```

C.2 owl:sameas construction

First using SPARQL to find all the triples

```
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#>
construct {
?sub owl:sameAs ?s.
?s owl:sameAs ?sub.
?s owl:sameAs ?o.
}
WHERE {
  ?s <http://example.com/pokemon/name> ?name.
  service <https://api.demo.triply.cc/datasets/academy/pokemon/services/pokemon/
  ?sub owl:sameAs ?o;
       rdfs:label ?name.
  }
}
```

Then using Python to merge them

```
import rdflib
```

```
from rdflib import Graph, ConjunctiveGraph, Literal, BNode, Namespace, RDF, URIR
from rdflib.namespace import DC, FOAF
import pprint
g1 = Graph()
g1.parse("queryResults.ttl", format="ttl")
#print(g1.serialize(format='nt'))
len(g1)
g2 = Graph()
g2.parse("statements1.nt", format="nt")
#print(g2.serialize(format='nt'))
len(g2)
g=g1+g2
len(g)
g.serialize(destination='new_pokemon.nt', format='nt')
g.serialize(format='nt')
```

C.3 Query

All the query code can be found in `https://krr.triply.cc/lafengzhenyu/datasetgifquery/sparql/datasetgifquery`, I just paste the 1V1 battle code here.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX pokemon: <http://example.com/pokemon/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT *{
{
    select ?aname ?atype ?aAttack ?agif ?aaAttack ?aHP1 ?aDefense ?aaDefense ?aI
  ?x pokemon:name ?aname . FILTER (contains (?aname, "Machamp"))
  ?x pokemon:hasType ?atype .
  ?x pokemon:attack ?aAttack .
  ?x pokemon:hp ?aHP1 .
  ?x pokemon:gif ?agif .
  ?x pokemon:defense ?aDefense .
  ?x foaf:depiction ?aImg .
      bind(round(xsd:integer(?aAttack)*0.08) as ?aaAttack)
      bind(round(xsd:integer(?aDefense)*0.02) as ?aaDefense)
    }
    limit 1
  }
  {
    select ?bname ?btype  ?bAttack ?bgif ?bbAttack ?bHP1
?bDefense ?bbDefense ?bImg
```

```
    where{
  ?x pokemon:name ?bname . FILTER (contains (?bname, "Blastoise"))
  ?x pokemon:hasType ?btype .
  ?x pokemon:attack ?bAttack .
  ?x pokemon:hp ?bHP1 .
      ?x pokemon:gif ?bgif .
  ?x pokemon:defense ?bDefense .
  ?x foaf:depiction ?bImg .
      bind(round(xsd:integer(?bAttack)*0.08) as ?bbAttack)
      bind(round(xsd:integer(?bDefense)*0.02) as ?bbDefense)
    }
    limit 1
  }

  bind(xsd:integer(?bHP1)-?aaAttack+?bbDefense as ?bHP2)
  bind(xsd:integer(?aHP1)-?bbAttack+?aaDefense as ?aHP2)
  bind(if(?aHP2>?bHP2,"wins","loses") as ?label1)
  bind(if(?aHP2>?bHP2,"loses","wins") as ?label2)
   bind('''
       <table>
<tr style="font-size: 200%; text-align: center; text-transform: uppercase;"><td>
          <td>
            <img src="{{agif}}" width="150" style="transform: scale(-1,1);">
          </td><td>VS</td>
          <td>
            <img src="{{bgif}}" width="150">
          </td>
        </tr>

        <tr style="font-size: 300%; text-align: center; text-transform: upperca
      </table>
<table border="0" cellpadding="1" style="font-size: 120%;">

        <tr><td style="width: 120px;">{{aAttack}} </td><th>Attack</td><td> 
        <tr><td>{{aDefense}} </td><th>Defense</td><td>  {{bDefense}} </
<tr><td>{{aHP2}} ({{aHP1}})</td><th>HP(original)</td><td>  {{bHP2}} ({{b
      </table>
       ''' as ?widget)
}
```