Bastien GUILLOU
Nicolas HAMEAU

Rapport : Approches sémantiques en Intelligence artificielle

Sommaire :

Introduction :

Ce rapport contient l'ensemble des réflexions liées à l'utilisation de l'IA symbolique pour aborder des problématiques liées aux combats dans le jeu vidéo pokémon. Ce rapport est lié au notebook, représentant tout le code qui a permis de répondre à cette problématique (ainsi pour toute capture du code du notebook, vous pouvez vous y référer directement). Nous commencerons par introduire notre sujet d'étude en présentant le jeu vidéo et son système de combat. Puis nous introduirons le concept de l'IA symbolique pour étudier ses principes, son fonctionnement et ses avantages. Cette IA symbolique doit utiliser un modèle de connaissance, qui sera présenté afin de regarder comment ces techniques de résolution de problème peuvent être appliquées. Et nous terminerons par une comparaison de l'IA sémantique à une autre méthode de machine learning, les réseaux de neurones, pour en trouver les avantages et inconvénients.

I. Le jeu pokémon:

Dans cette partie nous allons présenter le jeu vidéo pokémon afin de comprendre son système central : les combats tour par tour. Ce type de combat laissant place à de grandes possibilités de stratégie possible, il est alors primordial de comprendre le fonctionnement du jeu.

Le très populaire jeu vidéo pokémon, issu de la licence Nintendo, a été créé par Satoshi Tajiri en 1995. Ce jeu a été inspiré initialement de par la passion dévorante pour la capture des insectes de son créateur. Ainsi dans ce jeu, vous incarnez un “dresseur de pokémon” se baladant dans l’univers de pokémon afin de conquérir toutes les arènes d’une région pour y collectionner tous les badges disponibles, prouvant la puissance du joueur.

Ici une image des 8 badges d'arènes disponibles d’une région

Les pokémons sont des créatures imaginaires qui vivent en harmonie avec les humains. Chaque pokémon rencontré dans la nature, lors de l’aventure du joueur, peut être ainsi capturé afin de compléter l’équipe et la collection de pokémon qu’il possède.

Chaque joueur possède la possibilité de posséder entre 1 à 6 pokémons pour former son équipe de combat. Ainsi le but du jeu est de parcourir une région donnée afin de devenir le meilleur combattant pokémon. ceux en affrontant les meilleurs dresseurs de chaque arène et en se confrontant à d’autres dresseurs adverses qui peuvent être des personnages du jeu ou à des joueurs en ligne.

Ici une image d'une équipe de pokémon d'un dresseur

Les pokémons sont définis par :

- Un nom
- Des statistiques : PV, Attaque, Défense, Attaque Spécial, Défense Spéciale, Vitesse et Spécial
- Des types (entre 1 à 2 types) : "Normal", "Plante", "Feu", "Eau", "Electrik", "Glace", "Combat", "Poison", "Sol", "Vol", "Psy", "Insecte", "Roche", "Spectre", "Dragon", "Tenebres", "Acier", "Fee"
- Des valeurs pour chaque sensibilités aux types (la valeur variant suivant chaque association de types existants) agissant comme un coefficient multiplicateur sur les dégâts que peut subir le pokémon.
    - Une valeur de 0.5 symbolise une résistance à ce type à ce type d'attaque
    - Une valeur de 1 est neutre face à ce type à ce type d'attaque
    - Une valeur de 2 montre une sensibilité à ce type à ce type d'attaque
    - Une valeur de 4 montre une double sensibilité à ce type d'attaque

- Des attaques : Il peut posséder jusqu'à 4 attaques différentes ayant chacune des caractéristiques :
    - Un nom
    - Un type (suivant les types du pokémons il possédera un nombre limité de types d'attaque possible) : "Normal", "Plante", "Feu", "Eau",

"Electrik", "Glace", "Combat", "Poison", "Sol", "Vol", "Psy", "Insecte", "Roche", "Spectre", "Dragon", "Tenebres", "Acier", "Fee"

- Une catégorie :
  - Physique (basés sur les statistiques normales : Attaque et Défense)
  - Statut (non lié au statistique car donnant un effet au pokémon allié)
  - Spécial (basé sur les statistiques spéciales : Attaques Spécial et Défense Spécial)
- Une puissance
- Une précision
- Et un nombre d'utilisation possible par combat (PP)

II. Les combats pokémon :

De par cette approche simplifiée des pokémons, nous pouvons ainsi comprendre les mécanismes des combats. Il faut alors réussir à utiliser un pokémon avec des types non vulnérables aux attaques ennemies. Puis utiliser les attaques les plus puissantes de ce pokémon avec le type adapté afin d'exploiter les sensibilités des types du pokémon adverse.

Voici une mise en situation pour mettre au clair ces informations. Imaginons la situation dans laquelle vous possédez le pokémon Tortank.

№ 0009	Tortank

Artwork de Tortank pour *Pokémon Rouge Feu* et *Vert Feuille*.

Nom japonais	Kamekkusu カメックス *Kamex*
Nom anglais	Blastoise

Numéros de Pokédex régionaux			
Kanto	*Johto*$^{O\text{-}C}$	*Johto*^{HGSS}	*Hoenn*^{RSE}
009	234	239	211
Kalos (Centre)	*Isolarmure*		*Myrtille*
088	073		172

Navigateurs et autres numérotations			
Fiore	*Almia*	*Oblivia* (Présent)	*Oblivia* (Passé)
R-078	R-013	R-294	N-214
PokéPark	*Google Maps:* Pokémon Challenge	*Shuffle*	*Picross*
P-054	009	009	N21-08
Picross (Alternatif)		*Lentis*	
A21-03		100	

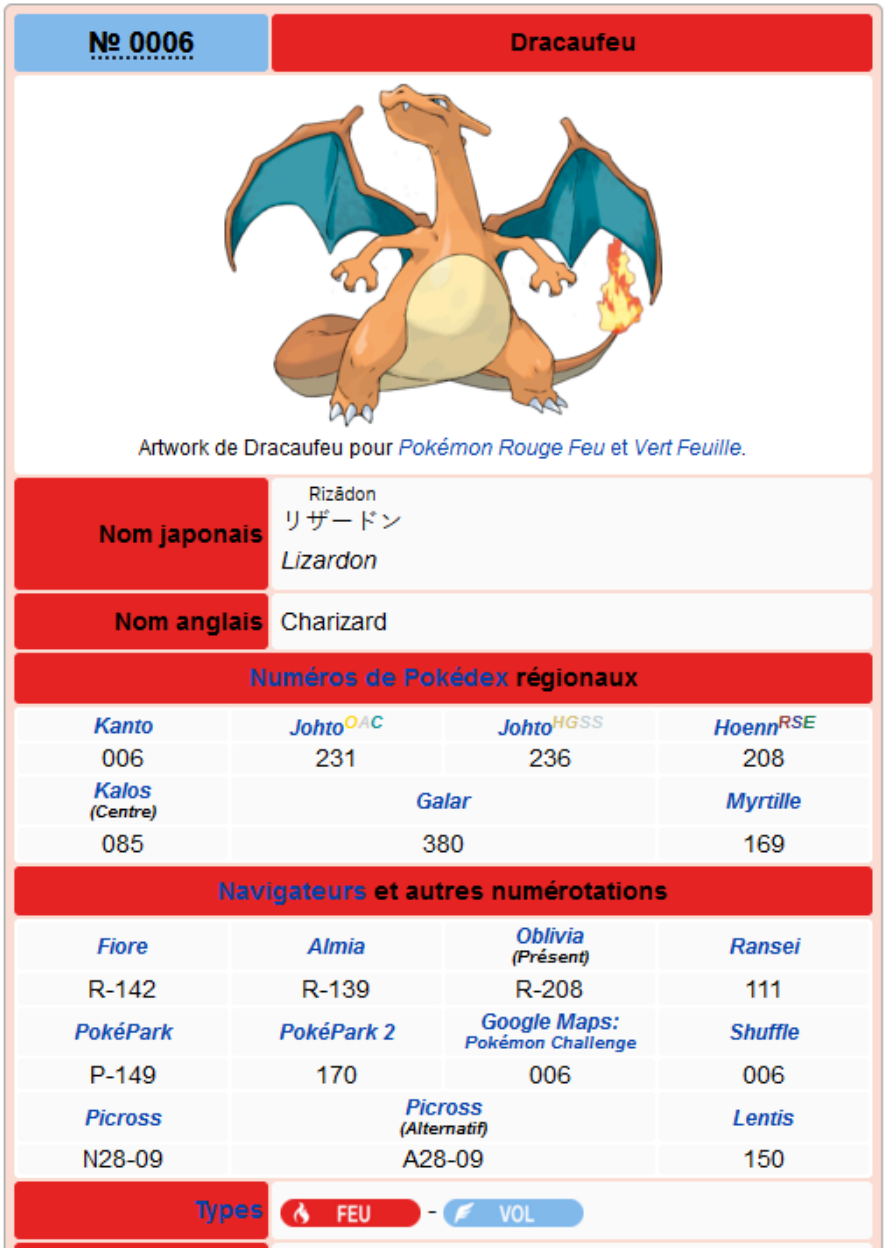
Type	EAU

Le pokémon tortank est un pokémon de type Eau

Sensibilités de Tortank																
Liste des sensibilités																
NORMAL	PLANTE	FEU	EAU	ÉLECTRIK	GLACE	COMBAT	POISON	SOL	VOL	PSY	INSECTE	ROCHE	SPECTRE	DRAGON	TÉNÈBRES	ACIER	FÉE
	× 2	× ½	× ½	× 2	× ½											× ½	
Légende	× 0 : Immunité		× ¼ : Double résistance			× ½ : Résistance			× 2 : Faiblesse			× 4 : Double faiblesse				

Voici la table de sensibilités des types du pokémon Tortank

Et vous combattez le pokémon adverse Dracaufeu.



№ 0006	Dracaufeu

Artwork de Dracaufeu pour *Pokémon Rouge Feu* et *Vert Feuille*.

Nom japonais	Rizādon リザードン *Lizardon*
Nom anglais	Charizard

Numéros de Pokédex régionaux

Kanto	*Johto* OAC	*Johto* HGSS	*Hoenn* RSE
006	231	236	208
Kalos (Centre)	*Galar*		*Myrtille*
085	380		169

Navigateurs et autres numérotations

Fiore	*Almia*	*Oblivia* (Présent)	*Ransei*
R-142	R-139	R-208	111
PokéPark	*PokéPark 2*	*Google Maps:* Pokémon Challenge	*Shuffle*
P-149	170	006	006
Picross	*Picross* (Alternatif)		*Lentis*
N28-09	A28-09		150

Types	FEU - VOL

Dracaufeu est un pokémon de type Feu et Vol

Sensibilités de Dracaufeu

Liste des sensibilités

NORMAL	PLANTE	FEU	EAU	ÉLECTRIK	GLACE	COMBAT	POISON	SOL	VOL	PSY	INSECTE	ROCHE	SPECTRE	DRAGON	TÉNÈBRES	ACIER	FÉE
	× ¼	× ½	× 2	× 2		× ½		× 0 [1]			× ¼	× 4				× ½	× ½

Légende	× 0 : Immunité	× ¼ : Double résistance	× ½ : Résistance	× 2 : Faiblesse	× 4 : Double faiblesse

Remarque(s)

• 1 : Si les attaques Gravité ou Anti-Air sont utilisées ou que Dracaufeu tient une Balle Fer, l'efficacité des attaques Sol à son encontre est double, du fait de son type Feu.

Voici la table des sensibilités des types du pokémon Dracaufeu

<u>Ainsi voici le scénario du combat simplifié :</u>

- Le pokémon tortank (de type Eau) est avantagé par rapport à Dracaufeu (de type Feu et Vol) car il possède des résistances aux types de Dracaufeu.
- De plus, vous constatez que Dracaufeu possède des sensibilités aux types de tortank et vous décidez d'utiliser une attaque de type eau pour profiter de cette faiblesse
- Vous pouvez choisir parmi les attaques que votre pokémon a apprises, une attaque de type eau, tel que Aquatacle qui est l'attaque la plus puissante dont vous disposez, afin de faire tomber les PV de Dracaufeu à 0.

Capacité						Niveau
Nom	Type	Catégorie	Puissance	Précision	PP	EV
[-] Masquer						
Luminocanon	ACIER		80	100 %	10	Évolution
Charge	NORMAL		40	100 %	35	Départ
Mimi-Queue	NORMAL		—	100 %	30	Départ
Pistolet à O	EAU		40	100 %	25	Départ
Repli	EAU		—	—	40	Départ
Tour Rapide	NORMAL		50	100 %	40	N.9
Morsure	TÉNÈBRES		60	100 %	25	N.12
Vibraqua	EAU		60	100 %	20	N.15
Abri	NORMAL		—	—	10	N.20
Danse Pluie	EAU		—	—	5	N.25
Hydro-Queue	EAU		90	90 %	10	N.30
Exuviation	NORMAL		—	—	15	N.35
Mur de Fer	ACIER		—	—	15	N.42
Hydrocanon	EAU		110	80 %	5	N.49
Aquatacle	EAU		120	100 %	10	N.56

Ici la liste des attaques possibles du pokémon Tortank

- De par ces choix vous gagnez le combat qui opposait ces 2 pokémons car vous avez utilisé le pokémon possédant des types avantageux contre le pokémon adverse. Vous avez utilisé une attaque puissante avec un type qui correspondait aux faiblesses du pokémon adverse.

III. La problématique des combats :

De par cette stratégie simplifiée, désormais vous serez en mesure de vous confronter à des challenges bien plus ardus car votre équipe peut avoir 6 pokémons différents tous avec des types différents ou similaires. Et vous pouvez affronter jusqu'à 6 pokémons adverses pour le même combat, sachant que chaque pokémon possède ses propres sensibilités qui sont liées directement à l'association de ses types.

C'est dans cette problématique bien précise que notre projet vient apporter une réponse afin de proposer aux joueurs un modèle de combat à suivre contre chaque pokémon adverse suivant les pokémons que vous possédez. Cette stratégie vise à utiliser les principales composantes d'un combat afin de proposer les scénarios de combat les plus adaptés contre chaque pokémon adverse.

IV. Introduction à l'IA symbolique :

Dans cette partie nous allons définir ce qu'est l'approche sémantique et comment elle peut être utilisée avec un raisonneur pour créer des requêtes qui répondent à notre problématique.

L'intelligence artificielle symbolique est souvent qualifiée d'IA basée sur la connaissance, qui désigne une méthode qui s'articule autour de symboles pour désigner les problèmes et leurs solutions grâce à la manipulation de ces symboles. Dans ce contexte, les symboles servent d'abstractions ou d'entités qui résument les propriétés de certains objets. Ce système de gestion de connaissances est appelé alors le réseau sémantique.

Par exemple, dans notre cas nous pouvons mettre en place un système de gestion des connaissances des pokémon où chaque pokémon possède des symboles représentés par ses types, attaques et sensibilités. Ces informations sont structurées de manière à permettre à notre IA symbolique de raisonner sur les interactions entre Pokémon, d'analyser leur compatibilité ou leur vulnérabilité, pour permettre ainsi de suggérer des stratégies de combat optimales en fonction des caractéristiques de l'adversaire.

V. Principes de l'IA symbolique :

Lors de la création de ce réseau sémantique, les systèmes d'IA symboliques fonctionnent grâce à l'utilisation de ses symboles et aux relations qui existent entre eux. Ces relations et connaissances peuvent être représentées par un graphe de connaissance ou une ontologie qui permettent donc de définir les relations existantes entre les entités. Par la suite, ces symboles peuvent être utilisés par des raisonneurs qui peuvent prendre des décisions basées sur les capacités de raisonnement, créer des plans pour des activités futures et même proposer de nouvelles hypothèses.

VI. Modélisation des connaissances :

Dans notre cas, l'ontologie a été formée sur la base des caractéristiques définissant un pokémon que sont ses types, ses attaques et ses sensibilités. Ainsi le raisonneur peut être utilisé avec des requêtes afin de questionner l'ontologie pour ressortir les informations demandées. Il est alors possible de formuler toutes sortes de requêtes à notre raisonneur à condition d'avoir établi correctement les liens qui existent entre les différentes entités afin que chaque instance puisse interagir entre elles.

Le premier pas dans l'application de notre IA symbolique au problème des combats de Pokémon a donc été de créer une ontologie. Cette ontologie nous a permis de définir les types d'entités (comme les Pokémon, leurs types, attaques, et sensibilités) ainsi que les relations entre ces entités. Par exemple, chaque Pokémon peut être relié à ses attaques, et chaque attaque peut être caractérisée par son type, sa puissance, et d'autres attributs pertinents. En utilisant un langage OWL (Web Ontology Language) de représentation des données avec owlready2 de python, nous avons construit une ontologie comprenant :

- Des classes telles que Pokemon, Type, Stat, Attack, et Sensi.
- Des propriétés d'objet comme hasType, hasAttack, hasAttackType, hasStat, et hasSensi, hasSensitype, valeur, qui permettent de relier les Pokémon à leurs types, attaques, statistiques et sensibilités respectives. Nous avons aussi utilisé nom qui permet de relier chaque caractéristiques d'un pokémon à son nom.
- Des propriétés de données pour décrire les attributs spécifiques de chaque entité, comme categorie, puissance, precision et pp pour les attaques ou valeur pour les valeurs de sensibilité d'un Pokémon à un certain type ou de ses statistiques.

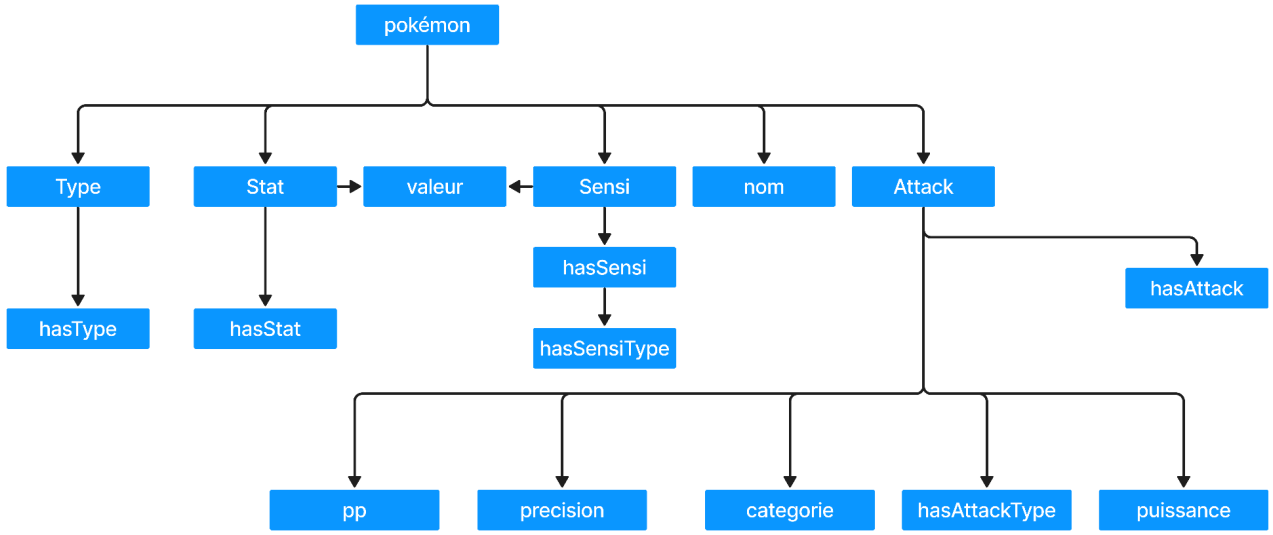
On obtient ainsi cette ontologie :

```
with onto:

    # Les classes caractérisant les pokémons
    class Pokemon(Thing):
        pass

    class Type(Thing):
        pass

    class Stat(Thing):
        pass

    class Sensi(Thing):
        pass

    class Attack(Thing):
        pass
```

```
# Type du pokémon
class hasType(ObjectProperty):
    domain = [Pokemon]
    range = [Type]

# Statistiques du pokémon
class hasStat(ObjectProperty):
    domain = [Pokemon]
    range = [Stat]

# Sensibilités du pokémon
class hasSensi(ObjectProperty):
    domain = [Pokemon]
    range = [Sensi]

# Attaques du pokémon
class hasAttack(ObjectProperty):
    domain = [Pokemon]
    range = [Attack]

# Nom du pokémon
class nom(DataProperty, FunctionalProperty):
    domain = [Pokemon, Type, Stat, Sensi, Attack]
    range = [str]

# Valeur de la statistique
class valeur(DataProperty, FunctionalProperty):
    domain = [Stat, Sensi]
    range = [float]
```

```python
# Type de sensibilité du pokémon
class hasSensiType(ObjectProperty, FunctionalProperty):
    domain = [Sensi]
    range = [Type]

# Type d'attaque du pokémon
class hasAttackType(ObjectProperty):
    domain = [Attack]
    range = [Type]

# Categorie de l'attaque
class categorie(DataProperty, FunctionalProperty):
    domain = [Attack]
    range = [str]

#  Puissance de l'attaque du pokémon
class puissance(DataProperty, FunctionalProperty):
    domain = [Attack]
    range = [int]

#  Precision de l'attaque du pokémon
class precision(DataProperty, FunctionalProperty):
    domain = [Attack]
    range = [int]

#  PP de l'attaque du pokémon
class pp(DataProperty, FunctionalProperty):
    domain = [Attack]
    range = [int]
```

Que l'on pourrait représenter sous forme d'ontologie :



VII. Raisonnement de l'IA symbolique :

Pour écrire nos requêtes, nous avons choisi d'utiliser SPARQL qui est un type de langage de requête spécialement conçu pour interroger les bases de données RDF (Resource Description Framework). Ainsi avec les requêtes SPARQL, nous pourrons obtenir des informations nécessaires à la réponse de notre problématique initiale à partir de notre ontologie. On peut par exemple réaliser des requêtes pour connaître les attaques les plus efficaces pour une certaine combinaison de type adversaire ou les vulnérabilités à certains types ciblés de pokémon.

Voici des exemples de requêtes que nous avons formuler à notre raisonneur :

Une requête permettant de lister tous les pokémons qui possède une sensibilité = 4 aux attaques d'un pokémon particulier :

```
def afficher_sensibilites_pokemon(reference_pokemon):
    query = f"""
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX onto: <http://example.org/onto/pokemon#>
    SELECT DISTINCT ?pokemonNom ?typeAttaqueNom ?valeurSensi
    WHERE {{
        ?pokemonAttaquant onto:nom "{reference_pokemon}" .
        ?pokemonAttaquant onto:hasAttack ?attack .
        ?attack onto:hasAttackType ?typeAttaque .
        ?typeAttaque onto:nom ?typeAttaqueNom .

        ?pokemonCible onto:hasSensi ?sensi .
        ?sensi onto:hasSensiType ?typeAttaque .
        ?sensi onto:valeur ?valeurSensi .
        FILTER(?valeurSensi = 4) .
        ?pokemonCible onto:nom ?pokemonNom .
    }}
    ORDER BY ?pokemonNom
    """

    results = default_world.sparql(query)
    print(f"Pokémons sensibles aux types d'attaques de {reference_pokemon} :")
    for result in results:
        pokemon_nom, type_attaque_nom, valeur_sensi = result
        print(f"- {pokemon_nom} est sensible au type {type_attaque_nom} avec une sensibilité de {valeur_sensi}.")

# Utilisation de la fonction pour "Dracaufeu"
afficher_sensibilites_pokemon("Dracaufeu")
```

```
Pokémons sensibles aux types d'attaques de Dracaufeu :
- Apireine est sensible au type Roche avec une sensibilité de 4.0.
- Apitrini est sensible au type Roche avec une sensibilité de 4.0.
- Artikodin est sensible au type Roche avec une sensibilité de 4.0.
- Bastiodon est sensible au type Combat avec une sensibilité de 4.0.
- Bastiodon est sensible au type Sol avec une sensibilité de 4.0.
- Blizzaroi est sensible au type Feu avec une sensibilité de 4.0.
- Blizzi est sensible au type Feu avec une sensibilité de 4.0.
- Cadoizo est sensible au type Roche avec une sensibilité de 4.0.
- Chapignon est sensible au type Vol avec une sensibilité de 4.0.
- Charmillon est sensible au type Roche avec une sensibilité de 4.0.
- Cheniselle est sensible au type Feu avec une sensibilité de 4.0.
- Cizayox est sensible au type Feu avec une sensibilité de 4.0.
- Coxy est sensible au type Roche avec une sensibilité de 4.0.
- Coxyclaque est sensible au type Roche avec une sensibilité de 4.0.
- Dimoret est sensible au type Combat avec une sensibilité de 4.0.
- Dinoclier est sensible au type Combat avec une sensibilité de 4.0.
- Dinoclier est sensible au type Sol avec une sensibilité de 4.0.
- Dracaufeu est sensible au type Roche avec une sensibilité de 4.0.
- Farfuret est sensible au type Combat avec une sensibilité de 4.0.
- Foretress est sensible au type Feu avec une sensibilité de 4.0.
- Galegon est sensible au type Combat avec une sensibilité de 4.0.
- Galegon est sensible au type Sol avec une sensibilité de 4.0.
- Galekid est sensible au type Combat avec une sensibilité de 4.0.
- Galekid est sensible au type Sol avec une sensibilité de 4.0.
...
- Tyranocif est sensible au type Combat avec une sensibilité de 4.0.
- Volcaropod est sensible au type Sol avec une sensibilité de 4.0.
- Yanma est sensible au type Roche avec une sensibilité de 4.0.
- Yanmega est sensible au type Roche avec une sensibilité de 4.0.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Requête permettant d'afficher la puissance et les types d'attaques d'un pokémon :

```python
# Requête SPARQL
query = """
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX onto: <http://example.org/onto/pokemon#>

SELECT DISTINCT ?attackName ?attackType ?power
WHERE {
  ?pokemon onto:nom "Dracaufeu" .
  ?pokemon onto:hasAttack ?attack .
  ?attack onto:nom ?attackName .
  ?attack onto:puissance ?power .
  ?attack onto:hasAttackType ?type .
  ?type onto:nom ?attackType
}
"""

# Exécuter la requête SPARQL
results = default_world.sparql(query)

# Afficher les résultats
for result in results:
    print(f"Nom de l'attaque: {result[0]}, Type: {result[1]}, Puissance: {result[2]}")
```

```
Nom de l'attaque: Lame d'Air, Type: Vol, Puissance: 75
Nom de l'attaque: Canicule, Type: Feu, Puissance: 95
Nom de l'attaque: Draco-Griffe, Type: Dragon, Puissance: 80
Nom de l'attaque: Griffe, Type: Normal, Puissance: 40
Nom de l'attaque: Rugissement, Type: Normal, Puissance: 0
Nom de l'attaque: Flammeche, Type: Feu, Puissance: 40
Nom de l'attaque: Brouillard, Type: Normal, Puissance: 0
Nom de l'attaque: Draco-Souffle, Type: Dragon, Puissance: 60
Nom de l'attaque: Crocs Feu, Type: Feu, Puissance: 65
Nom de l'attaque: Tranche, Type: Normal, Puissance: 70
Nom de l'attaque: Lance-Flammes, Type: Feu, Puissance: 90
Nom de l'attaque: Grimace, Type: Normal, Puissance: 0
Nom de l'attaque: Danse Flammes, Type: Feu, Puissance: 35
Nom de l'attaque: Feu d'Enfer, Type: Feu, Puissance: 100
Nom de l'attaque: Boutefeu, Type: Feu, Puissance: 120
Nom de l'attaque: Pouvoir Antique, Type: Roche, Puissance: 60
Nom de l'attaque: Cognobidon, Type: Normal, Puissance: 0
Nom de l'attaque: Morsure, Type: Tenebres, Puissance: 60
Nom de l'attaque: Riposte, Type: Combat, Puissance: 0
Nom de l'attaque: Draco-Charge, Type: Dragon, Puissance: 100
Nom de l'attaque: Draco-Queue, Type: Dragon, Puissance: 60
Nom de l'attaque: Griffe Acier, Type: Acier, Puissance: 50
Nom de l'attaque: Cru-Ailes, Type: Vol, Puissance: 60
Nom de l'attaque: Baston, Type: Tenebres, Puissance: 0
Nom de l'attaque: Colere, Type: Dragon, Puissance: 120
...
Nom de l'attaque: Rafale Feu, Type: Feu, Puissance: 150
Nom de l'attaque: Sable Ardent, Type: Sol, Puissance: 70
Nom de l'attaque: Rafale Ecailles, Type: Dragon, Puissance: 25
Nom de l'attaque: Double Volee, Type: Vol, Puissance: 40
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

VIII. La solution à notre problématique avec l'IA symbolique:

```python
# Liste des Pokémon du joueur
pokemons_joueur = ["Dracaufeu", "Dardargnan", "Dracolosse", "Raichu", "Entei", "Nosferalto"]

def meilleur_pokemon_contre(adversaire_nom, pokemons_joueur):
    pokemons_joueur_filtre = ', '.join([f'"{p}"' for p in pokemons_joueur])

    query = f"""
    PREFIX onto: <http://example.org/onto/pokemon#>
    SELECT DISTINCT ?pokemonNom ?attackName ?attackType ?power ?sensiValue
    WHERE {{
        # On récupère les noms et les attaques d'un pokémon afin de connaitre leurs types et leurs puissance et connaitre le type du pokém
        ?pokemon onto:nom ?pokemonNom .
        FILTER (?pokemonNom IN ({pokemons_joueur_filtre})) .
        ?pokemon onto:hasAttack ?attack .
        ?attack onto:nom ?attackName .
        ?attack onto:hasAttackType ?typeAttaque .
        ?typeAttaque onto:nom ?attackType .
        ?attack onto:puissance ?power .

        # On récupère les sensibilités du pokémons adverse et ses types afin de privilés les attaques efficace
        ?adversaire onto:nom "{adversaire_nom}" .
        ?adversaire onto:hasSensi ?sensiAdv .
        ?sensiAdv onto:hasSensiType ?typeAttaque .
        ?sensiAdv onto:valeur ?sensiValue .
        FILTER (?sensiValue >= 2) .

        # Vérifier que le Pokémon n'est pas vulnérable à aucun type de l'adversaire
        FILTER NOT EXISTS {{
            ?adversaire onto:hasType ?advType .
            ?pokemon onto:hasSensi ?pokemonSensi .
            ?pokemonSensi onto:hasSensiType ?advType .
            ?pokemonSensi onto:valeur ?sensiVuln .
            FILTER (?sensiVuln > 1) .
        }}
```

```
    }}

    # On tri nos résultats par ordre de puissance des attaques pour montrer les meilleures attaques
    ORDER BY DESC(?sensiValue) DESC(?power)

    # On affiche les 3 meilleurs résultats obtenu
    LIMIT 3
    """

    results = default_world.sparql(query)
    print(f"Contre {adversaire_nom}, les meilleures attaques à utiliser sont :")
    for result in results:
        pokemon_nom = result[0] if result[0] else "Inconnu"
        attack_name = result[1] if result[1] else "Inconnue"
        attack_type = result[2] if result[2] else "Type Inconnu"
        power = result[3] if result[3] else "Puissance Inconnue"
        sensiValue = result[4] if result[4] else "Efficacité Inconnue"
        print(f" - {pokemon_nom} avec l'attaque '{attack_name}' (Type: {attack_type}) ayant une puissance de {power} et une efficacité de

# Exécution de la fonction pour chaque adversaire
for adversaire in ["Lamantine", "Florizarre", "Hyporoi", "Pyroli", "Lippoutou", "Amonistar"]:
    meilleur_pokemon_contre(adversaire, pokemons_joueur)
```

```
Contre Lamantine, les meilleures attaques à utiliser sont :
 - Raichu avec l'attaque 'Electacle' (Type: Electrik) ayant une puissance de 120 et une efficacité de 2.0.
 - Raichu avec l'attaque 'Fatal-Foudre' (Type: Electrik) ayant une puissance de 110 et une efficacité de 2.0.
 - Raichu avec l'attaque 'Tonnerre' (Type: Electrik) ayant une puissance de 90 et une efficacité de 2.0.
Contre Florizarre, les meilleures attaques à utiliser sont :
 - Dracaufeu avec l'attaque 'Rafale Feu' (Type: Feu) ayant une puissance de 150 et une efficacité de 2.0.
 - Nosferalto avec l'attaque 'Rapace' (Type: Vol) ayant une puissance de 120 et une efficacité de 2.0.
 - Dracaufeu avec l'attaque 'Boutefeu' (Type: Feu) ayant une puissance de 120 et une efficacité de 2.0.
Contre Hyporoi, les meilleures attaques à utiliser sont :
 - Raichu avec l'attaque 'Calinerie' (Type: Fee) ayant une puissance de 90 et une efficacité de 2.0.
 - Raichu avec l'attaque 'Voix Enjoleuse' (Type: Fee) ayant une puissance de 40 et une efficacité de 2.0.
 - Raichu avec l'attaque 'Charme' (Type: Fee) ayant une puissance de Puissance Inconnue et une efficacité de 2.0.
Contre Pyroli, les meilleures attaques à utiliser sont :
 - Dracolosse avec l'attaque 'Hydro-Queue' (Type: Eau) ayant une puissance de 90 et une efficacité de 2.0.
 - Dracaufeu avec l'attaque 'Sable Ardent' (Type: Sol) ayant une puissance de 70 et une efficacité de 2.0.
 - Entei avec l'attaque 'Sable Ardent' (Type: Sol) ayant une puissance de 70 et une efficacité de 2.0.
Contre Lippoutou, les meilleures attaques à utiliser sont :
 - Dracaufeu avec l'attaque 'Rafale Feu' (Type: Feu) ayant une puissance de 150 et une efficacité de 2.0.
 - Dracaufeu avec l'attaque 'Boutefeu' (Type: Feu) ayant une puissance de 120 et une efficacité de 2.0.
 - Entei avec l'attaque 'Deflagration' (Type: Feu) ayant une puissance de 110 et une efficacité de 2.0.
Contre Amonistar, les meilleures attaques à utiliser sont :
 - Raichu avec l'attaque 'Electacle' (Type: Electrik) ayant une puissance de 120 et une efficacité de 2.0.
 - Raichu avec l'attaque 'Fatal-Foudre' (Type: Electrik) ayant une puissance de 110 et une efficacité de 2.0.
 - Raichu avec l'attaque 'Tonnerre' (Type: Electrik) ayant une puissance de 90 et une efficacité de 2.0.
```

Ainsi au travers de cette requête nous pouvons conseiller un joueur sur la stratégie à opter contre chaque adversaire en proposant les 3 meilleures attaques avec le pokémon associé qui réponde à ce problème. Avec des indications sur le nom du pokémon à utiliser, le nom de l'attaque, son type, sa puissance et l'efficacité de l'attaque. De cette manière, le joueur peut combattre avec une bonne stratégie d'attaque tout en essayant de protéger ses pokémons.

VIV. Les avantages et les limites de l'IA symbolique:

L'un des avantages majeurs de l'IA symbolique est qu'elle peut fournir des explications claires et explicites suivant les requêtes effectuées. Ce qui est indispensable pour des situations qui nécessitent une compréhension et une justification des décisions prises dans les jeux vidéo de stratégiques tels que pokémon ou les systèmes d'aide à la décision.

Toutefois, si les natures des données ne sont pas connues, cette approche peut présenter des limites. En effet, l'IA sémantique est incapable de créer de nouvelles connaissances qu'elle ne pourrait pas traiter correctement, dans des situations où les données sont incomplètes ou incertaines. Ce type d'approche nécessite une modélisation précise et complète des connaissances du domaine pour être efficiente. Ce qui tend à nous montrer les limites de cette approche, car d'autres approches en machine learning peuvent gérer ces situations bruitées et incomplètes de manière très efficiente.

Dans l'ensemble, l'utilisation de l'IA symbolique dans le domaine du jeu pokémon nous expose le potentiel qu'elle possède pour résoudre des problèmes complexes grâce à une compréhension bien développée de la situation.

X. Retour d'expérience de l'utilisation de l'IA symbolique:

Avantages de l'IA symbolique :

- **Explicabilité** : Fournit des explications claires et logiques lors des requêtes.
- **Formalisation des connaissances** : Permet de structurer et de formaliser explicitement les connaissances d'un domaine spécifique.
- **Manipulation des règles logiques** : Permettant de traiter des scénarios complexes avec précision.
- **Contrôle et sécurité** : Peu de risques d'erreurs lorsque les requêtes sont bien formulées.

Limites de l'IA symbolique:

- **Modélisation préalable** : Nécessite une modélisation détaillée et précise
- **Moins efficace avec des données incomplètes ou bruitées** : par exemple le fait qu'il n'existe pas de négation dans le raisonneur. Le raisonneur sait " ce qui n'est pas dans quelque chose" mais ne connaît pas "la négation pure" symbolisation par l'opposé de quelque chose.
- **Scalabilité** : problèmes de scalabilité lorsque la quantité de données ou leur complexité sont trop grandes. Lorsque nous ajoutons beaucoup de complexité dans les données, il est arrivé que le raisonneur ne comprenne

plus notre ontologie, bien qu'elle ait été correctement formée, car elle possédait trop de références pour éviter les multiples instances (faire une référence à chaque valeur existante sous la forme d'un dictionnaire pour éviter de créer de multiples instances pour la même donnée).

- **Flexibilité limitée** : Moins flexible pour s'adapter aux nouvelles situations qui pouvaient sortir des cadres de ses connaissances.
- **Dépendance à une expertise du domaine** : La construction d'ontologies et celles des requêtes demandent une attention très particulière, sans quoi rien ne pourrait fonctionner. Ce qui demande une certaine expertise dans le domaine ciblé pour éviter toute erreur dans les relations entre les entités de l'ontologie, sans quoi les requêtes ne pourraient aboutir.

XI. Comparaison de l'IA symbolique avec les réseaux de neurones :

- **Transparence contre Obscurité** :
  - **IA symbolique** : Offre une transparence élevée grâce à l'utilisation de règles et de logiques explicites.
  - **Réseaux de neurones** : Agit comme une boîte noire, rendant difficile les prédictions ou les décisions difficiles à comprendre.
- **Gestion des connaissances** :
  - **IA symbolique** : Nécessite une modélisation explicite des connaissances. Il faut ainsi une expertise du domaine pour établir correctement l'ontologie et les requêtes.
  - **Réseaux de neurones** : Peut apprendre de manière automatique sur de grandes quantités de données et trouver les relations importantes qui permettent de réaliser les prédictions. Facile à mettre en place, la seule étape difficile concerne la préparation des données pour le modèle.
- **Performance avec données limitées** :
  - **IA symbolique** : Peut fonctionner efficacement car elle ne dépend pas de la quantité des données entrées.
  - **Réseaux de neurones** : Nécessite de grandes quantités de données pour atteindre une performance optimale avec des soucis de performances importants (méthode souvent très coûteuse en calculs).
- **Adaptabilité** :
  - **IA symbolique** : Moins adaptable aux changements ou aux données non prévues.
  - **Réseaux de neurones** : Peut s'adapter et apprendre de nouvelles tendances avec de nouvelles données, ce qui l'aide à performer.
- **Complexité et coût de développement** :
  - **IA symbolique** : Le développement initial complexe et coûteux.
  - **Réseaux de neurones** : Collecte et préparation des données importantes mais sont accessibles à tous publics pour obtenir de bonnes performances.