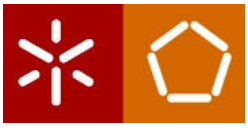


Universidade do Minho
Departamento de Informática

Perfil de Engenharia de Linguagens
Representação e Processamento de Conhecimento na Web

Trabalho Prático

Grupo 5

Gonçalo Costa - PG55944

José Correia - PG55967

Rodrigo Casal Novo - PG56006

Índice

1. Introdução

Este relatório documenta o desenvolvimento de um sistema baseado em ontologias, realizado no contexto da unidade curricular de Representação e Processamento de Conhecimento na Web (RPCW).

O principal objetivo deste projeto consistiu na conceção, implementação e gestão de uma ontologia orientada para o domínio do universo fictício Pokémon. Esta ontologia visa representar, de forma estruturada e semântica, o conhecimento associado a entidades típicas deste universo, como Pokémons, regiões, ataques, itens, tipos e relações entre estas entidades.

O sistema foi desenvolvido com base nos padrões de representação de conhecimento RDF (Resource Description Framework) e OWL (Web Ontology Language), permitindo uma modelação formal e extensível da informação. Estas tecnologias possibilitam não só a descrição de classes, propriedades e instâncias, como também a inferência automática de novo conhecimento através de raciocínio semântico.

Para além da definição da ontologia, foram também desenvolvidas consultas em SPARQL com o intuito de extrair conhecimento específico e relevante a partir dos dados modelados, demonstrando assim a utilidade prática da ontologia em tarefas de consulta e análise.

2. Tecnologias / Requisitos

- **Python 3.12+** — Linguagem principal para scripts, processamento de dados e povoamento da ontologia.
- **HTML/CSS/JavaScript** — Frontend da aplicação.
- **Flask** — Framework usado na interface da aplicação.
- **RDFlib** — Biblioteca Python para manipulação de grafos RDF e serialização.
- **Requests** — Biblioteca Python para realizar pedidos HTTP a APIs.
- **D3**[1] - A biblioteca de JavaScript para visualização de dados personalizada.
- **GraphDB** — Triplestore para armazenamento e consulta da ontologia via SPARQL.
- **APIs Externas** — PokeAPI, GraphDB e Kaggle, para coleta e persistência dos dados.

3. Estratégia de Desenvolvimento

3.1. Modelação da Ontologia

Inicialmente o grupo optou por procurar por datasets completos e consistentes no Kaggle[2] para conseguir formar um ontologia base simples e que service de ponto de partida. A partir da análise de dois Datasets encontrados, que foram combinados num só com recurso a um *join* (join.py), as primeiras classes, relações e atributos mais relevantes foram criados no protege, formando assim a nossa *pokemon_ontology_base.ttl*.

Após isso com recurso a livraria de python **RDFlib**[3], pois é a mais flexível e fácil de manter e atualizar, foi gerado o primeiro script de povoamento, *populate_script.py*, onde usando o CSV *final_pokemon.csv*, as primeiras instâncias de Pokemons, regiões, tipos, habilidades e os seus atributos e relações, foram criadas.

Insatisfeitos com o atual tamanho da ontologia, o grupo decidiu pesquisar por mais dados, onde encontramos uma Api com bastante informação relevante do universo de Pokemon, sendo essa a **PokeAPI**[4], com recurso à biblioteca **requests**[5] foi possível acedar a diversos endpoints, com

vários dados relevantes. Ainda numa segunda parte no *populate_script.py* através da mesma, foram povoados todos os itens existentes, assim como as relações e atributos convenientes e necessárias (*pokemon_new_povoada_withitems.ttl*) e num segundo *populate_script2* (para evitar estar a correr tudo de novo e aumentar a eficiência do processamento dos dados), foi usado para povoar todos os movimentos e golpes dos Pokémons (pokemon_new_povoada_withmoves.ttl). Sendo assim a nossa ontologia acabou com a seguinte estrutura:

3.1.1. Classes

- Ability
- ItemAttributes
- Items (e as suas várias subclasses)
- Moves
- Pokemon (com subclasse de normal, lendário e pseudo-lendário)
- Region
- Type

3.1.2. Object Properties

- effectiveness (e diversas subclasses e subsubclasses para melhor organização e compreensão)
- evolvesFrom (Pokemon - Pokemon)
- EvolvesTo (Pokemon - Pokemon)
- fromType (Move - Type)
- hasAbility (Pokemon - Ability)
- hasAtribute (Item - ItemAttributes)
- hasType (Pokemon - Type)
- isFrom (Pokemon - Region)
- learnby (Moves - Pokemon)

3.1.3. Data Properties

- Name
- ID
- Weight
- Height
- Class
- Power
- Effect

(...)

3.2. Persistência e Consulta

A persistência do conhecimento representado na ontologia foi assegurada com a integração com GraphDB, um repositório triplestore que permite armazenar, consultar e modificar grafos RDF. Decidimos criar uma instância para armazenar a ontologia, e desta forma garantir o armazenamento persistente dos dados.

A integração com GraphDB possibilita a execução eficiente de consultas SPARQL sobre a ontologia.

As consultas SPARQL foram implementadas para permitir a extração e manipulação eficiente dos dados da ontologia dos Pokémon. Utilizou-se a biblioteca SPARQLWrapper para comunicar com o endpoint do GraphDB, possibilitando tanto consultas de leitura (GET) como operações de escrita (POST), como INSERT, DELETE e UPDATE. Estas funcionalidades permitem a criação dinâmica de novas relações, a inserção de instâncias e propriedades, bem como a remoção ou alteração de dados existentes na ontologia garantindo escalabilidade.

Para manter a consistência e facilitar o acesso aos dados do repositório, é utilizado um ficheiro `config.json` onde são armazenadas e atualizadas as informações essenciais da instância atual como, o URL do servidor GraphDB, o nome do repositório ativo e o prefixo base da ontologia.

3.3. Povoamento Automático da Ontologia

Primeiro é criado um repositório no GraphDB. A criação automática do repositório é feita através de um pedido HTTP POST à sua API REST, usando um ficheiro de configuração Turtle (.ttl) que define o repositoryID, o tipo (graphdb:SailRepository), o conjunto de regras de inferência (rdfsplus-optimized), etc..

Após a criação, a ontologia é carregada com recurso à biblioteca RDFlib, que lê o ficheiro Turtle, serializa os dados e envia os triplos RDF para o endpoint via POST. Os dados ficam imediatamente disponíveis para consulta e manipulação através de queries SPARQL.

3.4. Desenvolvimento da Aplicação Web

Foi desenvolvida uma interface web intuitiva utilizando o framework Flask, com o objetivo de permitir aos utilizadores explorar, pesquisar e consultar a ontologia de forma acessível e interativa. A aplicação disponibiliza funcionalidades para a importação e exportação de ontologias, bem como a execução de queries SPARQL personalizadas, possibilitando a extração e análise de conhecimento de acordo com os critérios definidos pelo utilizador.

Adicionalmente, foi implementado um módulo de visualização gráfica, que permite representar as relações entre entidades da ontologia em formatos interativos, facilitando a compreensão da estrutura e dos vínculos semânticos do conhecimento representado.

4. Funcionalidades

4.1. Gestão da Ontologia

Criar e gerir a estrutura da ontologia. Assim como, definir classes, propriedades e relações entre entidades.

4.2. Execução de Consultas SPARQL

Executar consultas SPARQL para extrair e manipular dados da ontologia.

4.3. Pesquisa de Entidades

Pesquisar entidades por nome.

4.4. Persistência de Dados

Ontologias armazenadas num repositório do GraphDB.

4.5. Upload e Download de ficheiros .ttl

Permite carregar um ficheiro RDF para o repositório e fazer download da ontologia atual, através de pedidos POST e GET, respetivamente, ao endpoint dos triplos (`/repositories/{repo}/statements`) do repositório. Sendo possível fazer o download da ontologia em diferentes instâncias.

4.6. Eliminar Ontologia

Limpa completamente o repositório especificado, removendo o mesmo das instâncias do GraphDB, através de um pedido DELETE ao endpoint do repositório (`/repositories/{repo}`).

5. Páginas

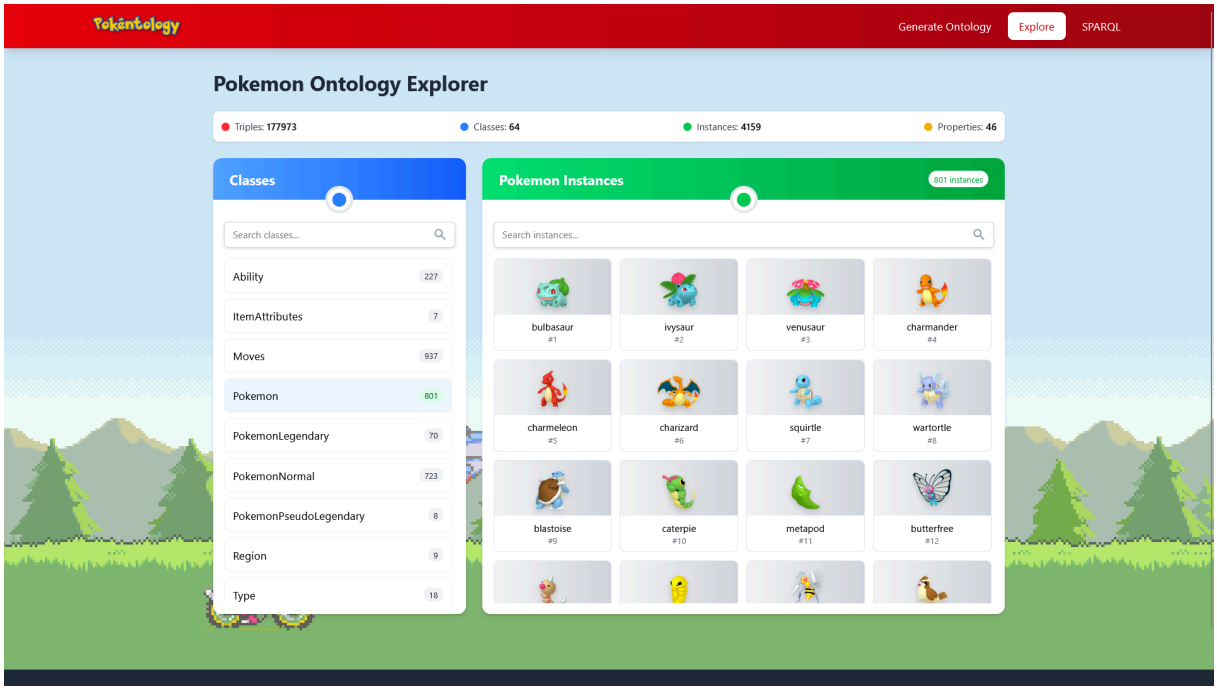
Abaixo estão presentes descrições breves de algumas páginas desenvolvidas, e algumas acompanhadas de uma captura de ecrã:

5.1. Página Inicial

Página inicial que sugere ao utilizador explorar a ontologia.

5.2. Página de Exploração

Explora a ontologia, por classes e entidades. Também é possível ver estatísticas atuais da ontologia e aplicar filtros de pesquisa.



5.3. Página da Entidade

Ver informações detalhadas sobre uma entidade específica, incluindo as suas propriedades e relações com outras entidades.

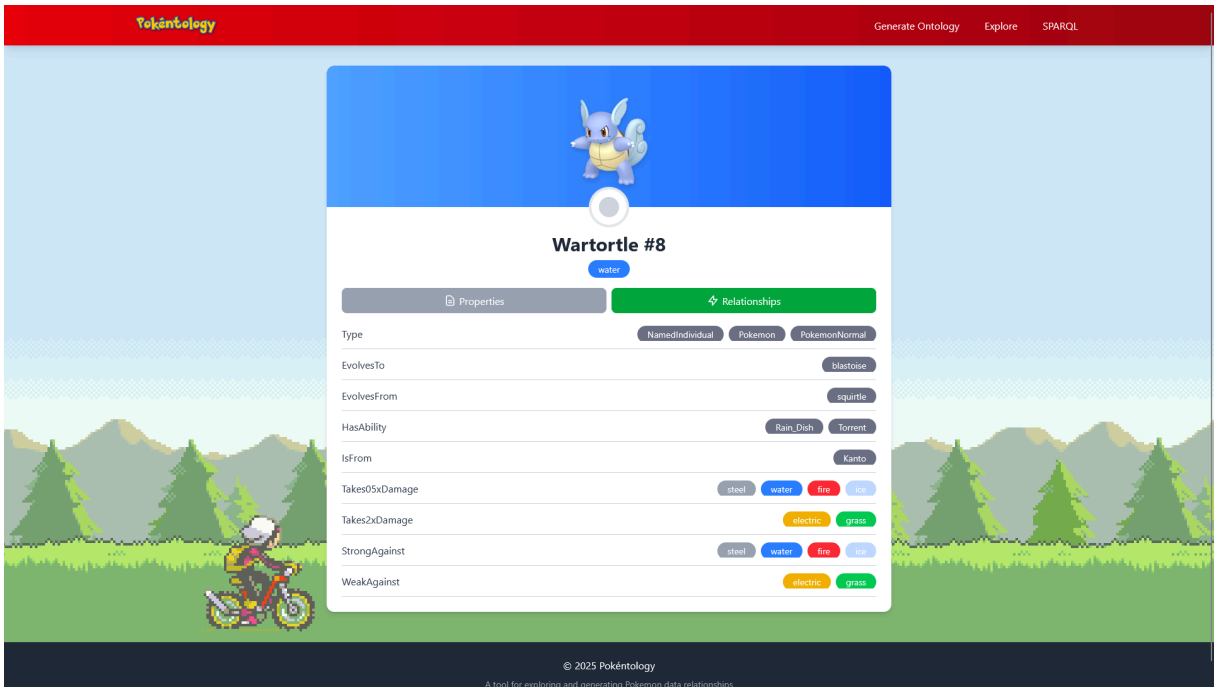


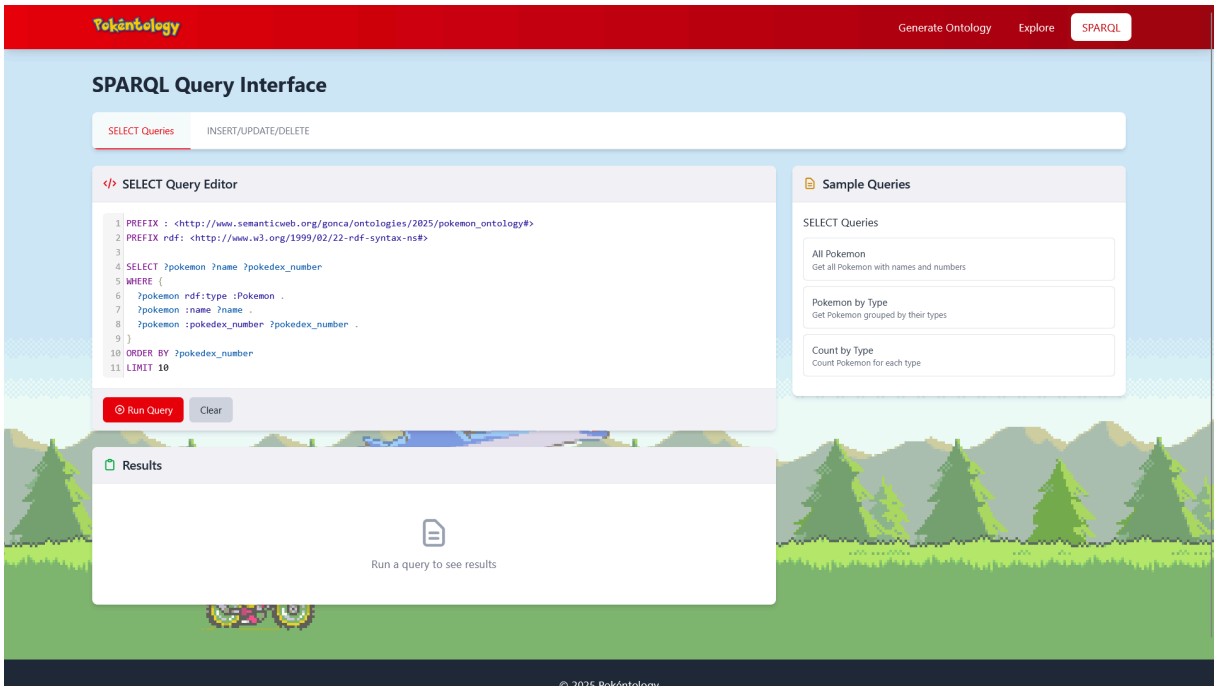
Figura 2: Relações

5.4. Página da Ontologia

O utilizador aqui pode gerar uma nova ontologia, no fundo reiniciar o processo. E também tem a capacidade de carregar uma ontologia a partir de um ficheiro '.ttl', transferir a ontologia atual ou eliminar a mesma.

5.5. Página de Consultas e Edições SPARQL

Permite explorar a ontologia com queries de consulta SPARQL.



5.6. Visualiação em grafo

Permite o utilizador analisar a ontologia numa perspetiva visual, que muitas vezes pode ajudar a encontrar relações complexas entre entidades, implementado com recurso à biblioteca D3.

Apesar do esforço significativo por parte do grupo na tentativa de implementar esta funcionalidade da forma mais eficiente possível, a sua concretização ficou limitada devido à elevada exigência computacional associada. As máquinas utilizadas durante o desenvolvimento não dispunham dos recursos necessários para suportar o processamento intensivo requerido, o que comprometeu o desempenho e a viabilidade da funcionalidade em questão, não sendo possível testar a mesma e alcançar resultados mais satisfatórios.

6. Correr projeto

Na root do projeto correr:

```
python3 run.py
```

Sendo necessário ter um container com o GraphDB atualmente a correr e todas as dependências anteriormente ditas instaladas.

7. Conclusão

O desenvolvimento deste sistema permitiu aplicar, de forma prática, os conceitos abordados na unidade curricular, promovendo a compreensão das tecnologias semânticas, bem como das boas práticas de modelação ontológica.

A partir de um desenvolvimento mais aprofundado, foi possível perceber ainda melhor as vantagens reais de ontologias, nomeadamente na capacidade de estruturar e interligar conhecimento de forma explícita e reutilizável, facilitando a interoperabilidade entre sistemas, a automatização de inferências lógicas e a consistência na representação de informação complexa. Este tipo de abordagem revela-se particularmente útil em domínios ricos em entidades e relações, como é o caso do universo Pokémon, onde a diversidade de elementos e as interdependências entre os mesmos beneficiam significativamente de uma representação semântica clara e formalizada.

De trabalho futuro, nós não pretendemos que a aplicação "morra" por aqui, querendo acrescentar ainda mais elementos, como Treinadores, a opção de criar a sua própria equipa e automaticamente perceber as vantagens e fraquezas da mesma, com um sistema de recomendação automático, entre outras, e eventualmente fazer "deploy" à mesma.

Bibliografia

[1] Observable, «The JavaScript library for bespoke data visualization». [Online]. Disponível em: https://d3js.org/

[2] Anthony Goldbloom, «Level up with the largest AI & ML community». 2025.

[3] Daniel Krech, «RDFLib - Read the Docs». [Online]. Disponível em: https://rdflib.readthedocs.io/en/stable/

[4] Poké API, «The RESTful Pokémon API». [Online]. Disponível em: https://pokeapi.co/

[5] Python, «Requests: HTTP for Humans». [Online]. Disponível em: https://requests.readthedocs.io/en/latest/