ISEL

DEETC - Mestrado em Engenharia Informática

Guia Aula Prática

Representação e Processamento de Conhecimento (RPC)

---

1. Suporte programático para o conceito de "Linked Data"

Vamos usar o ambiente SPARQLWrapper que recorre ao RDFLib e permite interagir com os "SPARQL-endpoint". Vamos explorar o "endpoint" RDF4J (da aula anterior) e outros na Web.

a) A versão do SPARQLWrapper que usaremos está em `distribuicao_03` (é versão 1.8.1).

Se pretender descarregar diretamente vá a: `https://rdflib.github.io/sparqlwrapper`

b) Para instalar descomprimir `sparqlwrapper-*.zip`, iniciar ambiente de consola, mudar para a pasta `sparqlwrapper-*` e executar: `python setup.py install`

Documentação em: `https://rdflib.github.io/sparqlwrapper/doc/latest/.`

2. Testar "SPARQLWrapper" e analisar o JSON-LD

a) Verifique a instalação executando `a01_SPARQLendpoint_DBPedia_A.py`. Note que deve obter uma mensagem de erro (ou de "Warning") relativo às utilização do JSON-LD. Comente a linha "`import rdflib_jsonld`" e volte a executar; agora não deve obter erro.

b) O JSON-LD (JSON for Linked Data) é o formato mais recente para transferência de informação, por um "SPAQRL-endpoint", no contexto da "Linked Data".

Aceda a `http://json-ld.org/` e analise o enquadramento do JSON-LD.

c) O JSON-LD tem uma implementação, `rdflib-jsonld`, que funciona como "plug-in" do RDFLib. Vamos usar essa implementação.

A versão do `rdflib-jsonld` que usaremos está em `distribuicao_03` (é a versão 0.4).

Pode descarregar diretamente em: `https://pypi.python.org/pypi/rdflib-jsonld`.

d) Para instalar descomprimir `rdflib-jsonld-*.zip`, iniciar ambiente de consola, mudar para a pasta `rdflib-jsonld-*` e executar: `python setup.py install`

e) Para verificar a instalação descomente a linha "`FROM SPARQLWrapper import JSONLD`" e execute `a01_SPARQLendpoint_DBPedia_A.py`. Agora já não deve obter mensagem de erro. Caso mantenha a mensagem de erro é provável que a nova instalação ainda não tenha sido "incorporada" no ambiente de desenvolvimento (e.g., IDLE); se assim for pode re-iniciar o ambiente de desenvolvimento.

3. Testar diretiva SPARQL por interação com "SPARQL-endpoint"

a) Aponte o seu Browser para `http://dbpedia.org/sparql` e copie para essa página a diretiva SPARQL que está em `a01_SPARQLendpoint_DBPedia_A.py`. Execute (nessa página) a diretiva e note que obtém uma resposta em várias Línguas.

b) Altere (diretamente na página Web) a diretiva SPARQL de modo a excluir as Línguas Japonês ("ja"), Chinês ("zh"), Russo ("ru") e Árabe ("ar"). *Sugestão*: `lang(?label)="ja"` filtra Japonês.

c) Altere a diretiva em `a01_SPARQLendpoint_DBPedia_A.py` para ficar igual à da alínea anterior.

4. O "SPARQLWrapper" e os formatos JSON e XML

Considere o código em `a01_SPARQLendpoint_DBPedia_A.py`

a) Elimine os comentários referentes ao pedido de resposta em JSON e analise os resultados.

b) Sobre os comentários referentes ao pedido de resposta em XML comece por eliminar apenas os comentários até "`print( resultSetXML )`" (inclusive); execute e analise o resultado.

c) Elimine agora os comentários do ciclo e analise o resultado.

d) Comente agora as 2 linhas abaixo de "`assert XML...`" e analise o resultado.

e) Para entender melhor como funciona o SPARQLWrapper pode analisar o código fonte que é bastante pequeno e legível; está em `(...)/SPARQLWrapper/Wrapper.py`.

5. ... o formato N3, conversão e desenho do Grafo

Considere o código em `a01_SPARQLendpoint_DBPedia_A.py`

a) Elimine os comentários referentes ao pedido de resposta em N3 e analise os resultados (para aumentar a legibilidade pode comentar o código relativo a formatos analisados na alínea anterior)

b) Elimine os comentários relativos à construção do grafo e analise os resultados.

c) Desenhe, no papel, o grafo gerado pela resposta da diretiva SPARQL. Note que este grafo não é "uma parte" do grafo original mas apenas uma descrição da resposta à interrogação SPARQL.

6. Formato JSON-LD e "fallback" para o reconhecido pelo "endpoint"

Considere `a01_SPARQLendpoint_DBPedia_A.py`. Na sequência dos exercícios anteriores vamos verificar que nem todos os "SPARQL-endpoint" respondem em todos os possíveis formatos.

a) Elimine os comentários referentes ao pedido de resposta em JSON-LD. Note que apenas aqui estamos a incluir o JSONLD para evidenciar que a mensagem de erro, tratada no início desta aula, teve origem do SPARQLWrapper (e não no nosso código).

b) Depois de analisar este código complete a função, no ficheiro `myENDPOINT_access.py`, que:

- aceita, como "input", uma diretiva SPARQL, uma lista com os formatos a usar pela ordem de prioridade de "fallback" e um "SPARQL-endpoint", e
- devolve, como "output", um tuplo com 2 elementos, onde o primeiro indica o formato em que a resposta foi obtida e o segundo contém a resposta efetivamente obtida (nesse formato).

7. ... estrutura JSON e transformação em "lista-de-listas"

Considere o código em `x_util_JSON.py` e `a02_SPARQLendpoint_DBPedia_B.py`

a) Analise `x_util_JSON.py` e note a estrutura JSON na qual é recebida uma resposta SPARQL. As variáveis "`x`" e "`Concept`" teriam sido obtidas via "`SELECT ?x ?Concept`" numa diretiva SPARQL. No código copie todo o JSON associado a `JSONvar` para `<ALTERAR>` e altere esse código de modo a substituir as variáveis "`x`" e "`Concept`", respectivamente por "`myX`" e "`myY`".

b) Adicione a variável "`myZ`" acrescentando os dados que considerar necessário.

c) Execute o código e verifique que as três variáveis estão, de facto, a ser apresentadas; deve obter um "output" com o formato: `valor-de-myX || valor-de-myY || valor-de-myZ`. Note que precisa de ajustar o código que imprime o "output" de modo a adicionar os valores de "`myZ`".

d) Analise `a02_SPARQLendpoint_DBPedia_B.py` e note o modo como a estrutura JSON é convertida. Na primeira conversão a extração das variáveis devolvidas na interrogação está "hard-coded". A segunda conversão usa o código da alínea anterior que itera na lista de variáveis na estrutura JSON construindo uma lista-de-listas (que pode ser manipulada como tabela).

e) Estenda o código que desenvolveu em `myENDPOINT_access.py` de modo a converter em lista-de-listas uma interrogação que seja devolvida em formato JSON.

8. "Linked Data" – exploração (programática) do repositório RDF4J

a) Analise e execute `b01_SPARQLendpoint_RDF4J.py`. Atenção: o RDF4J precisa de estar disponível (Servlet Jetty) para conseguir obter resposta à interrogação.

b) Note que, no código que está a analisar, a variável "`repository`" tem o nome do repositório que irá interrogar (no RDF4J). O repositório indicado no código refere-se àquele que construiu na aula prática anterior. No entanto pode colocar qualquer outro mas não se esqueça de ajustar a interrogação em conformidade!

c) Vá eliminado os comentários no código de modo a apresentar o resultado nos vários formatos.

9. Aplicação para explorar a "Linked Data"

a) A partir dos exemplos anteriores e de todo o trabalho que já efetuou construa uma aplicação (simples) que aceite como "input" um ficheiro (pode ser um ficheiro Python) com uma interrogação SPARQL e que gere, como "output" um outro ficheiro com uma representação RDF (em formato XML ou outro) do grafo resultante dessa interrogação.

b) A interrogação deve ser feita a um "SPARQL endpoint" indicado pelo utilizador; por exemplo, via outro ficheiro de "input" ou via parâmetro em linha de comando ou por interação com o utilizador.