Engenharia de Dados e Conhecimento

universidade de aveiro

Trabalho Prático 2

2017/2018

I WanTo Know

António Sérgio Oliveira Silva
76678 asergio@ua.pt

1. Introdução ao Tema

O tema deste trabalho é idêntico ao do primeiro.

Porém, com a introdução da Web Semântica neste trabalho, foi necessário aplicar alterações na sua estrutura interna. De maneira a dar uso à Web Semântica existe a necessidade de haver relações entre os vários dados, algo que, no primeiro trabalho é praticamente inexistente.

A base de dados no primeiro trabalho era formada apenas por um conjunto de funcionalidades da aplicação e os inputs do utilizador correspondente a cada uma. Existe relação, mas não o suficiente para explorar a Semântica.

Foi então adicionada à base de dados deste projeto os resultados correspondentes a cada input. Desta maneira já foi possivel obter uma rede semântica suficientemente completa para investigação e exploração das várias tecnologias usadas.

2. Dados, suas fontes e sua transformação

Nada mudou nas suas fontes em relação ao primeiro trabalho, continua a ser usada a API do Wolfram|Alpha e o Reddit Atom para apresentação do Feed.

Contudo, é importante anotar que estas também não passam por nenhuma nova transformação !

Porquê ? Ao realizar este trabalho o foco foi sempre explorar ao máximo as tecnologias e, por isso, foi necessário “fugir” do comum. Os dados devolvidos pela API do Wolfram|Alpha são muito simples para ser usada uma transformação para RDF.

Então, não foi usada transformações para RDF nas fontes de dados, mas sim na transformação da base de dados do primeiro trabalho (XML) para RDF de maneira a ser usada no GraphDB já preparada com as alterações referidas na introdução.

Versão **simplificada** do XSL usado:

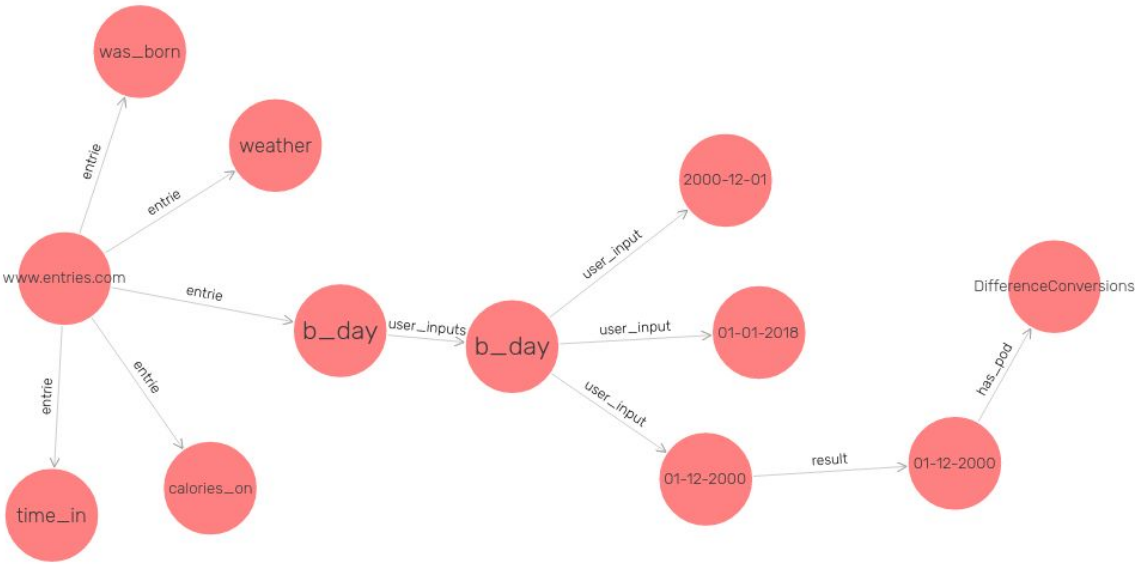
```
<pred:title>  </pred:title>
<pred:action> </pred:action>
<pred:user_inputs>
      <xsl:for-each select="user_inputs/user_input">
            <pred:user_input>
                        <pred:input></pred:input>
                        <pred:times></pred:times>
                        <pred:result></pred:result>
            </pred:user_input>
      </xsl:for-each>
</pred:user_input>
```

No elemento com predicado ‘result’ vai conter o resultado para o dado input.

3. Operações sobre os dados

Houve uma grande exploração do SPARQL neste projeto.
Foi usado para gestão da base de dados e para adquirir dados através da Wikidata.

Sendo a aplicação totalmente dependente da base de dados, existe uma grande variedade de queries usadas.



Com a introdução de armazenar os resultados na base de dados, houve a necessidade de adicionar queries que:

- Verifique se existe um resultado para um dado input e se este é válido (um resultado pode expirar) - **exists_result()**
- Que devolva o resultado armazenado para um dado input - **get_result()**
- Armazene o resultado obtido - **store_result()**
- Caso o resultado já tenha expirado, que o apague - **delete_old_result()**

As restantes queries são equivalentes às existentes no primeiro trabalho, apenas convertidas para SPARQL.

- Armazenar o input do utilizador - **store_user_input()**
- Actualizar vezes que o input foi referido - **update_times()**
- Obter vezes que o input for referido - **get_current_value()**
- Verificar se um input existe - **check_if_exists()**
- **etc.**

Exemplo de query usado para exists_result()

```
query = """
               PREFIX pred: <http://www.entries.com/pred/>

                     ASK{
                         $d pred:action \"""+action+"""\".
                         $d pred:user_inputs $user_input.
                         $user_input pred:user_input $c.
                         $c pred:input \"""+ user_input +"""\".
                         $c pred:result $r.
                         $r pred:has_pod $p.
                         $r pred:expires $timestamp.
                         FILTER($timestamp > """+ str(int(time.time())) +""")
                     }
               """
```

Notar que no fim existe um filtro para verificar se o resultado ainda é válido. Isto é feito verificando se o timestamp atual já ultrapassou o timestamp de “expires” do resultado.

Além da base de dados, o SPARQL também é usado para obter dados da Wikidata. Isto será referido mais à frente no relatório.

4. Publicação de dados semânticos

A publicação de dados semânticos e RDFa foram, sem dúvida, a parte que gerou mais confusão.

Foi difícil, para alguém que não participou nas aulas, entender o conceito. Porém, com a leitura da documentação do W3 (https://www.w3.org/TR/rdfa-primer/) foi possivel entender o objetivo e a sua importância.

Como lhe dar uso corretamente, já não é tão fácil de entender!

Contudo, foi incluída no trabalho, neste caso, nas templates onde os resultados vão ser apresentados.

Exemplo do template de b_day:

```
<div class="time_in-results" xmlns:pred= "http://www.entries.com/pred/"
                             about= "http://www.entries.com/result/b_day/{{user_input}}"
                             typeof= "pred:result">

<h1> You were born... </h1>

<span rel="pred:has_pod" href="http://www.entries.com/pod/b_day/{{user_input}}/DifferenceConversions">

        <h3 property="pred:content" > {{results.0}} </h3>

        <h3 property="pred:content" > {{results.1}} </h3>

        <h3 property="pred:content" > {{results.2}} </h3>

        <h3 property="pred:content" > {{results.3}} </h3>

</span>

</div>
```

Foi usando um span contendo todos os elementos de resultado pelo que, todos tinham como sujeito o:

'.../pod/b_day/{{user_input}}/DifferenceConversions'

Foi usado o 'rel' no spam para marcar a relação que existe entre o:

- **'.../result/b_day/{{user_input}}**

  e
- **'.../pod/b_day/{{user_input}}/DifferenceConversions'**

5. A integração de dados da Wikidata

Como referido anteriormente, é usada a Wikidata para obtenção de dados com SPARQL.

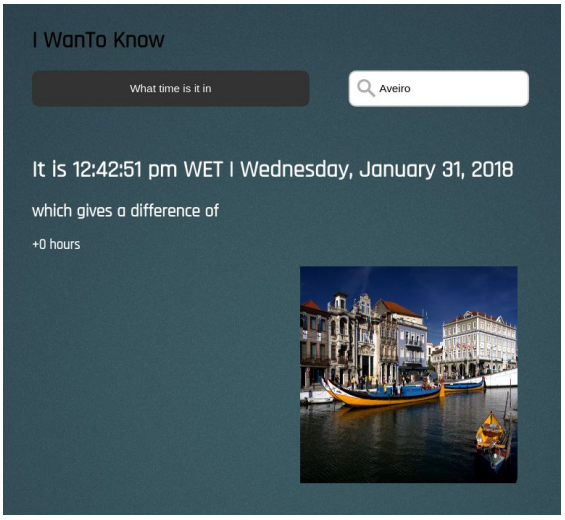
Neste projeto existem duas funções responsáveis para tal.

- **get_person_pic() -** dada o nome de uma pessoa, devolve uma foto e uma descrição da mesma se existente.
- **get_pic_location() -** dada o nome de uma localização, devolve uma foto da mesma se existente.

Query para get_pic_location():

```
query = """
            SELECT ?pic
            WHERE {
                    ?item wdt:P31/wdt:P279* wd:Q486972.
                    ?item ?something \"""+location+"""\".
                    OPTIONAL { ?item wdt:P18 ?pic. }
                  }
      """
```

A sua aplicação:



6. Funcionalidades da Aplicação

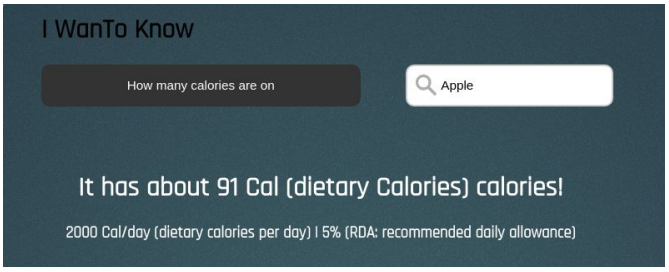
As funcionalidades da aplicação são também equivalentes à primeira parte, nomeadamente:

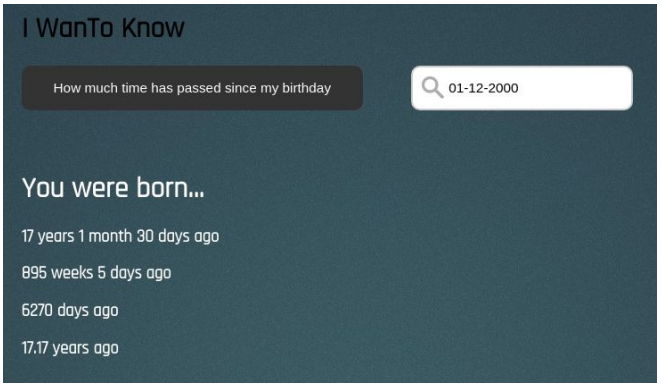
Com esta aplicação é capaz de:

- Conhecer as últimas notícias sobre o Wolfram
- Saber quanto dias passaram desde determinada data
- Saber que pessoas notáveis nasceram em determinada data
- Saber as horas actuais em dado local
- Saber que calorias se encontram em dado alimento
- Saber a temperatura atual em dado local

A diferença com a primeira parte é apenas a rapidez do output para resultados já armazenados e a apresentação de imagens das cidades pesquisadas / pessoas referidas.

Imagens das funcionalidades:



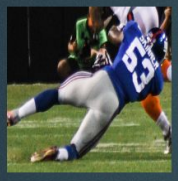


I WanTo Know

Who was born in

18-12-1990

People Born in this day:

Deontae Skinner	Hiroko Kuwata	Jarrod Pughsley	Micah Johnson	Shin-Ae Ahn	Sierra Kusterbeck
		Photo Not Available	Photo Not Available	Photo Not Available	Photo Not Available
American football player	Japanese tennis player				

I WanTo Know

How's the weather in

Moscow

Something between -15 °C and -4 °C snow (late afternoon to late night)

7. Conclusões

Não é novidade nenhuma que dar a um computador uma aproximação do saber humano traz resultados incríveis. A Web Semântica é apenas mais uma prova.

O que mais impressionou durante o desenvolvimento deste trabalho foi a aplicação da Wikidata. A capacidade de obter tal quantidade de informação numa questão de meia dúzia de linhas traz uma nova visão do que podem vir a ser os dados no futuro da Internet.

Impressionante, ou talvez assustador, mas imagino que acima de tudo, vamos ver aplicações incríveis em diferentes áreas de aplicação como a inteligência artificial.

8. Configuração para Executar a Aplicação (Linux)

A aplicação é dependente da base de dados!

É necessário executar o GraphDB e ter nele um repositorio com o nome “entries” onde deve importar o ficheiro : `/app/static/database/entries.rdf`

Para iniciar o GraphDB (na pasta /opt/GraphDBFree):
`./GraphDBFree &`

Para criar o repositório e importar o ficheiro pode fazê-lo na interface gráfica acedendo a :

http://127.0.0.1:7200/

A interface gráfica e muito user-friendly pelo não deverá ter problemas em fazê-lo, certifique-se apenas que import do ficheiro é no repositório “entries”.

Quanto a packages python, necessita das packages django,lxml, wikidata, rdflib e s4api. Caso não possua alguma pode facilmente as instalar com:
`pip3 install lxml django wikidata s4api rdflib`

Por fim, é necessário iniciar a framework (na pasta raiz): `python manage.py runserver`